Personalized Outfit Recommendation System

Kaavish Report
presented to the academic faculty
by

Muhammad Ali	ma02526
Osama Yousuf	oy02945
Syeda Areeba Kazmi	sk02901
Tasneem Adnan	ta02903

In partial fulfillment of the requirements for
Bachelor of Science
Computer Science

Dhanani School of Science and Engineering

Habib University
Spring 2020

Personalized Outfit Recommendation System

This Kaavish project was supervised by:

Dr. Shahid Hussain
Faculty of Computer Science
Habib University

Approved by the Faculty of Computer Science on _____.

Dedication

To our families...
For their support, their faith and their patience.

Acknowledgements

We would like to express our sincere gratitude to our supervisor, Dr. Shahid Hussain, for his support and guidance.

We would also like to thank our Kaavish Committee and the panelist evaluators: Dr. Saleha Raza, Dr. Waqar Saleem, Dr. Mussabir Majeed, and Dr. Abdul Samad, Dr. Taj Khan, Dr. Umair Azfar Khan, and Ms. Nadia Nasir for their insightful comments, questions, and critique. Deepest gratitude to Dr. Saleha Raza and Dr. Mussabir Majeed for guiding us in the initial steps of our thesis.

Our sincere thanks to our fellow batchmates: Ahsan Ahmed, Asad Osman, Burhanuddin Hakimi, Fazl Ur Rehman, Hasan Shakir, Hunain Raza, Jahanzaib Hyder, Mahad Ali, Minhaj Ahmed Moin, Muhammad Ali, Muhammad Haris Nadeem, Muhammad Tahir, Muneeb Bin Shoaib, Mursalin Larik, Nisha Sheikh, Rohan Ali, Saad Saleem, Sahran Riaz, Sameer Anees, Sami Murtaza, Sanie Hanfi, Syed Hamza Ali, Syed Hamza Azeem, Syed Meisum, Syed Sameer Nadeem, Talha Javed, Umair Hashmi, Wahaj Ahmad, and Zain Rahat for participating in the fashion photoshoot for our dummy-brand on our web-application.

Lastly, we would like to thank each other and our families for understanding us in difficult times and constantly supporting us morally throughout the year.

Abstract

The two main challenges faced by outfit recommendation is of visual understanding and matching. It includes the extraction of visual features and mixing and matching them according to the features compatibility and human capability and fashion sense. This can be time-consuming and cumbersome when done manually. Even though retailers spend an ample amount of time in selecting and organizing the products to provide the best combinations of outfits to the consumers, the options are still not personalized.

To address this problem, we propose a personalized outfit recommendation system (PORS). PORS is a web-application for consumers, which allows a shopper to search a product visually by uploading an image as an input and get similar (and of the same type e.g. a shirt for a shirt) and complementary (e.g. pants for a shirt) outfit recommendations from the items available on the store.

PORS uses a model, which uses an approach relying only on a feature network and use Euclidean distance to find similar vectors. Our feature network consists of a ResNet50 model trained on the DeepFashion - the largest and best-annotated dataset in the domain of fashion consisting of more than 800,000 richly annotated images, with 50 categories and 1000 descriptive attributes - clothing classification benchmark using transfer learning. This feature network converts images into an extensive feature vector. Images whose feature vector has minimum Euclidean distance from the query feature vector; are then displayed on our web-application.

Currently, some of the local stores like J., Furor, Zellbury, Export Leftovers, and a dummy store, are available on our web-application. The model was further trained on the stores available on our web-application but as the eastern dataset was comparatively less, so our model suffers from underfitting. However, the performance would improve as we train our model on some more eastern clothing.

Contents

List of Figures

List of Tables

1. Introduction

This chapter discusses the main idea of the project, describing the problem and its proposed solution, highlighting the target audience for whom the solution is being worked upon and some of the limitations which might affect the pace of work.

1.1 Problem Statement

Domain: Fashion e-commerce.

Facts and Figures:

1. Personalized shopping is the future of commerce. It is reported that on average today, at least 27% of retail site revenue in fashion, which totals to around USD 870 million, comes from personalized recommendations systems. [1]

2. In 2013, over 85% of Amazon sales revenue came through personalized recommendations. [2]

3. Despite all its potential, the Pakistani fashion industry is lagging behind in keeping up with such advances in personalized recommendation systems. [3]

Statement: One of the biggest problems in fashion retail is product curation. Retailers have to spend a large amount of time to come up with different combinations of their products that would as a whole, go well as an outfit, and even then, the clothe choices aren't really personalized and according to one's fashion sense. A customer buys a new shirt, brings it home, and hangs it up, only to find that the shirt stays in their closet for weeks because they're not sure what to pair it with. This also means a loss in conversion rates and potential revenue at the side of the retailer.

1.2 Proposed Solution

As already described, fashion retailers spend a lot of time manually curating their products, and according to a report published by *emerj.com* (database of reports on AI technology), at least 40% of potential revenues are lost because of poor outfit recommendations. We see a business opportunity in this problem, and so the idea behind the project is to solve it by addressing the key issue, product curation, by providing expert recommendations across different clothing items to the end-consumer at the point of sale or as a standalone service.

Our solution is a web application that would allow shoppers to visually search the catalogue of e-commerce stores by uploading pictures of outfits they like or taking a photo with their phone's camera. Using Computer Vision, the outfit would be broken down into its constituent parts (eg. shirt, pants, belt, sneakers) and identical and/or visually similar items from the store would be shown at the same place. This would allow shoppers to quickly and conveniently shop for items they see on social media, significantly increasing conversion rate.

1.3 Intended User

According to a recent study, millennials and Generation Z are the most coveted demographics for e-commerce stores. They do 60% of their shopping online [4] and make more apparel purchases than other generations [5]. On average, they spend three hours per day on their phones, mostly on social media platforms such as Facebook and Instagram, constantly consuming and interacting with visual content.

Our intended users are these consumers of visual content, and in order to appeal to them, it is essential for e-commerce stores to change the way shoppers interact with their stores. When someone sees their favourite Instagram influencer wearing an outfit that they want, searching for each piece of that outfit via text is not only cumbersome, it is inefficient and unlikely to yield accurate results. In order to allow customers to shop the same way they interact with social media i.e. via images, fashion e-commerce stores are increasingly looking to Artificial Intelligence and Computer Vision powered solutions.

To ensure practicality and applicability, we have been gathering and incorporating feedback from HU faculty as well as industry professionals from Love For Data, Daraz.pk, and PCSIR.

Our application would primarily provide two sets of recommendations when an item is being viewed by a user:

1. Items **visually similar** (and of the same type e.g. shirt for shirt) to that currently being viewed, increasing the likelihood that shoppers will find an item they like that is available in their size and at an agreeable price point.

2. Items **visually complementary** to that being viewed, allowing users to "Complete the Look". This allows stores to upsell and increase Average Order Value (AOV).

In addition, we will also actively look into personalized fashion recommendations based on user purchase history and general trends.

1.4 Key Challenges

A few key challenges that we have identified to foresee in this project are listed below, along with possible ways to address them.

1. We require a dedicated machine in one of the University's labs for hosting our web-server and preferably also a web hosting service. A possible remedy is to take use of local hosting. However, it must be noted that this would increase difficulty in collaborating.

2. Similarly, unavailability of a GPU can hinder the precision of the recommendation system, which would be created entirely from scratch. A simple remedy is to resort to cloud-based services for GPUs such as AWS or Google CoLab.

3. Another challenge would be to clean and curate the dataset as per our requirements and domain. Pre-existing datasets (explained further in chapter 3) may not be exactly in a usable condition out-of-the-box. Therefore, the data would then need to be scraped and cleaned manually which can be cumbersome. A remedy would be to maintain a clean storage format from the get-go.

4. In addition to this, lack of relevant technical knowledge on part of the team is also a challenge. This will be addressed by taking tutorials and online courses.

5. At the same time, insufficient knowledge and expertise in the domain of e-commerce requires us to reach out to industrial partners and professionals from Daraz and Telemart, whose unavailability at times can obstruct the smooth progression of our project.

2. Literature Review

Recommendation systems have become a common source for attracting audiences [1]. Various websites recommend a variety of products to their users, such as books, movies, songs, etc. But these products can't be separated, while there are products which can be separated into different parts and categories, like outfits. As online shopping and fashion-focused social networks are growing rapidly, there is a great need for intelligent fashion recommendation. Retailers have to spend a large amount of time to come up with different combinations of their products that would as a whole, go well as an outfit, and even then, the options aren't really personalized.

Hence, our project will have two main components, a web application and a recommendation system. Therefore, our literature review is divided into two parts.

2.1 Web-Application

The literature review for our web-application is divided into three domains. Technologies used for client-side, server-side and the API for establishing the interaction between them.

Table 2.1, Table 2.2, and Table 2.3 offer a detailed comparison between different communication interfaces.

Server-side Technologies			
Django	**Tornado**	**ASP.NET**	**Node.JS**
In Python, good support for ML out of the box.	Also In Python, good support for ML out of the box.	In C#, Visual Basic, F#, high dependency on external ecosystems for ML, not as intuitive.	In JavaScript, moderate support for ML out of the box.
High community support.	Low community support.	Moderate community support.	High community support.
Not completely asynchronous, though offers support.	Highly synchronous.	Supports both, synchronous as well as asynchronous operations.	Supports both, synchronous as well as asynchronous operations.
Supports direct integration with React/Front-end frameworks/AJAX	Third-party libraries for integration.	Completely different eco-system	Direct integration with React and Angular.
Offers abstraction, is high level.	Also high level.	Relatively low-level.	Relatively low-level.

Table 2.1: Technology Comparison: Server-side Communication [6] [7]

Client-side Technologies		
Django	**React**	**Angular**
Bound to a Django server, imposes design and flow restrictions.	Not bound to a specific server, gives high design freedom.	Not bound to a specific server, gives moderate design freedom.
Complete framework, potential overheads.	Only a library on top of JS, low overheads.	Complete framework, potential overheads.
Low community support.	High community support.	Moderate community support.
Offers high throughput in development.	Offers high throughput performance-wise.	Offers high throughput performance-wise.

Table 2.2: Technology Comparison: Client-side Communication [8]

API Technologies	
REST	**GraphQL**
Every resource is identified by a unique URL having its own end-point that requires router handlers.	Resources are identified by fields in their schema representation with individual resolvers.
Each request calls exactly one route handler, for a nested query, this leads to overheads.	Each query can call many resolvers to construct a nested response with multiple resources.
The response shape has to be manually constructed and is kept fixed.	The response shape is constructed automatically to match the query shape.
As a result, resource type, shape, & fetch query are coupled.	Everything is de-coupled leading to higher scalability.

Table 2.3: Technology Comparison: API-level Communication [9] [10]

Our Approach:
Based on the project requirements and keeping the communication channels in mind, the chosen technology stack for the client-end is: TypeScript, React, and React Apollo (for interacting with the GraphQL API), and for the server-end is: Python, Django, and Graphene Django (for creating the GraphQL API).

2.2 Recommendation System

This literature review regarding recommendation system, aims to compare the various methods and techniques studied in the existing work for recommendation systems. It is divided into two domains, the dataset and recommendation engine.

2.2.1 Dataset

The largest and best annotated publicly available dataset in the domain of fashion is **DeepFashion** [11].

The DeepFashion dataset consists of more than 800,000 richly annotated images, ranging from well-posed shop images to unconstrained consumer photos, making it twice the size of the largest previously available dataset. Each image in the dataset is labeled with 50 categories, 1,000 descriptive attributes, and clothing landmarks.

Another strength of the DeepFashion dataset is that it contains rigorous benchmark for testing the performance of algorithms for clothes recognition. The three benchmarks it contains are clothing attribute prediction, in-shop clothes retrieval,

and cross-domain clothes retrieval, a.k.a. street-to-shop. All three are relevant for the scope

1. **Category and Attribute Annotation:** Each image is labelled with a single category label. There are a total of 50 mutually exclusive categories, labelled by human annotators. The dataset also contains 1000 attributes which are extracted from image metadata. However, these are not utilized by our model. There are 63,720 diverse images in this benchmark.

2. **In-Shop Clothes Retrieval:** This task determines if two in-shop images belong to the same clothing item. This benchmark contains 54,632 images of 11,735 clothing items.

3. **Consumer-to-Shop Clothes Retrieval:** Aimed at matching consumer-taken photos with their shop counterparts. Contains 251,361 consumer-to-shop image pairs.

	DCSA [3]	ACWS [1]	WTBI [12]	DDAN [4]	DARN [10]	**DeepFashion**
# images	1856	145,718	78,958	341,021	182,780	**>800,000**
# categories + attributes	26	15	11	67	179	**1,050**
# exact pairs	N/A	N/A	39,479	N/A	91,390	**>300,000**
localization	N/A	N/A	bbox	N/A	N/A	**4∼8 landmarks**
public availability	yes	yes	no	no	no	**yes**

Figure 2.1: Comparing DeepFashion with other existing datasets. DeepFashion offers the largest number of images and annotations.

Our Approach
We will be using DeepFashion dataset as it contains large number of outfit images which are vastly categorized and labelled to train our model and benchmark performance. We will then train the model further on our local dataset, scraped from local stores like, Export Leftovers, J. Furor, Zellbury, etc. As eastern data is comparatively less, our model would suffer from overfitting if used as the only source.

2.2.2 Recommendation Engine

Viet et al. [12] learned featured transformation for measuring compatibility between different pairs of items using a Siamese CNN architecture, whereas, McAuley et al.[13] used parametric distance transformation for the same purpose. Parametric

distance transformation assigns the lowest distance to pairs of clothing which fit well. However, these techniques only gave the matching pairs of clothing and didn't take the personalization issue into account. The initial attempt to explore the personalized outfit recommendation was made by Hu et al. [14], using functional tensor factorization method, however, they used hand-crafted features.

Recently, researchers have explored deep networks and have begun to apply deep learning to recommendation systems. Table 2.4, Table 2.5, and Table 2.6 gives a detailed comparison between different approaches used in multiple research papers.

Recommendation Engine				
Research Paper	**Approach**	**Dataset**	**Resources**	**Time**
Large Scale Visual Recommendations From Street Fashion Images.	Recommends visually complementary items using Deterministic Fashion Recommenders (DFR) and Stochastic Fashion Recommender.	Fashion-136K, Fashion-350K, Fashion-Q1K datasets.	Not mentioned.	Not mentioned.

Table 2.4: Technology Comparison: Recommendation Engine

Recommendation Engine				
Research Paper	**Approach**	**Dataset**	**Resources**	**Time**
Learning visual similarity for product design with convolutional neural networks.	Training with stochastic gradient descent. t-SNE algorithm to visualize the result.	Dataset made from Houzz.com.	MTurk to collect the necessary bounding boxes. AlexNet to detect both near and exact duplicates.	Using a Grid K520 GPU it took 100 ms to compute.
MatchNet: Unifying Feature and Metric Learning for Patch-Based Matching.	Deep convolutional network extracts features and a network of three FC layers computes similarity. Feature network is influenced by AlexNet. ReLU for the convolution layers.	UBC patch dataset.	AlexNet	18 hours to 1 week to train the full network.
FashionNet: Personalized Outfit Recommendation with Deep Neural Network.	VGGNet feature network for feature extraction and multilayer fully connected network for computing clothes compatibility.	Dataset collected from Polyvore	VGGNet. Caffe.	Not mentioned.

Table 2.5: Technology Comparison: Recommendation Engine

Recommendation Engine				
Research Paper	**Approach**	**Dataset**	**Resources**	**Time**
DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations.	FashionNet simultaneously predicts landmarks and attributes. Network structure similar to VGG-16 except last convolutional layer, which is replaced by three branches of layers. Regression loss for landmark localization. Softmax loss for the predictions of categories. Triplet loss for metric learning	800, 000 diverse fashion images ranging from well-posed shop images to unconstrained consumer photos. Mogujie, Forever21 and Google Images	FashionNet to demonstrate usefulness.	Not mentioned.
Using Very Deep Autoencoders for Content-Based Image Retrieval	Train RBM by standard contrastive divergence learning procedure. To reduce noise, we use the probabilities rather than the stochastic binary states. Fine Tune auto-encoder by back propogation.	Preprocessed 1.6 million 32×32 color images. CIFAR-10 dataset.	Restricted Boltzmann Machines.	2 days on Nvidia GTX 285 GPU

Table 2.6: Technology Comparison: Recommendation Engine

Our Approach

The paper *MatchNet: Unifying Feature and Metric Learning for Patch-Based Matching* [15], published by Google Research in 2015 served as the inspiration for the FashionNet architecture proposed by the creators of DeepFashion. They use a single network to simultaneously predict both landmarks and attributes. The structure of this network is similar to VGG-16, with the last convolutional layer replaced by three branches of layers carefully designed for clothes.

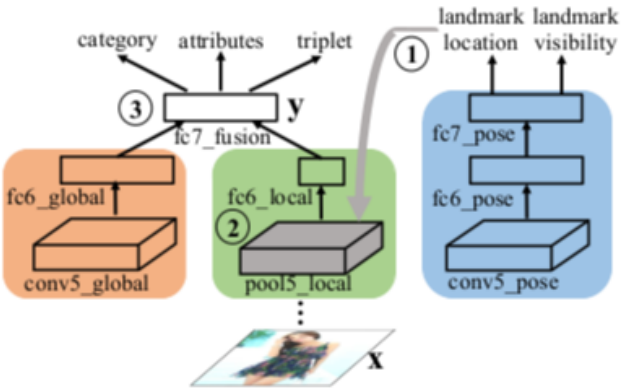

Figure 2.2: Pipeline of FashionNet, which consists of global appearance branch (in orange), local appearance branch (in green) and pose branch (in blue). Shared convolution layers are omitted for clarity. [11]

Since the only task we wish to perform in this project is in-shop retrieval and in the interest of making our model simpler, we therefore decided to use an approach relying on only a feature network and use Euclidean distance to find similar vectors. Our feature network will consist of a ResNet50 model trained on the DeepFashion clothing classification benchmark using transfer learning. This feature network will convert images into a feature vector. We will then query the database to retrieve images who's feature vector has minimum euclidean distance from the query feature vector.

3. Software Requirement Specification (SRS)

This chapter aims to provide detailed specifications of the system under development, including the functional and non-functional requirements, giving an overview of the application by providing the mock-up screens along with the details of the interface. This chapter also sheds light upon the working of certain features of the application.

3.1 Functional Requirements

This section describes each function/feature provided by our system. These functions are logically grouped into modules based on their purpose/users/mode of operations etc (as per our system):

3.1.1 Web Application

Some of the functionalities provided by the we application are as follows:

- Allows customers to upload a photo of an outfit.
- Displays constituent items of uploaded outfit.
- Allows customers to click on a constituent item and view items identical/similar to it.
- Allows customers to click on an item and open it's product page.
- Displays item title, description, price, photos and sizes on product page.
- On the product page, displays recommendations for similar products, as received from the recommendation engine.

- On the product page, displays recommendations for complementary products, as received from the recommendation engine.
- Allows customer to add item to cart.
- Allows new customers to sign-up.
- Allows returning customers to login.
- Saves user history to database.
- For existing customers with user history, the application displays recommendations based on user history, as received from the recommendation engine, on the home page.
- For existing customers with a user history, the application displays recommendations based on user history, as received from the recommendation engine, on product pages.
- For new customers or existing customers without user history, the application displays items currently trending on the store, on the home page.
- Admin has exclusive access to an admin panel that allows them to add, remove and modify items on the website.
- Admin can view customer activity.

3.1.2 Recommendation Engine

The engine can be roughly compartmentalized as follows:

- On uploaded photo, calls feature extraction model and performs multiple object detection. To bounding boxes of each individual piece of clothing in the outfit.
- Crops each bounding box in image and returns them. Each bounding box represents an article of clothing.
- Upon receiving which bounding box has been selected by the user, it passes that to nearest neighbour model. Returns ids of n closest neighbours of that item from the database and ids of n best complimentary items.

3.1.3 Database

- Should store id, title, price, description, sizes, category and tags for each item.
- Should store user and admin account information.
- Should store user profile information, including past logs, search history, purchase history, as well as other related actions.

3.1.4 Web Scraper

- Crawl local fashion stores 'Furor', 'Export Leftovers', 'J.', 'Zellbury', etc.
- For each item of men's clothing on the store, save the photo, description and price locally.
- Add timed cron job functionality so that the database can be verified and kept up to date without discrepancies.

3.2 Non-functional Requirements

Some of the non-functional requirements of the system under development are identified as follows:

3.2.1 Performance Requirements

- High performance of the computer on which the server is hosted is needed to cater to thousands of users.
- Fetching the dashboard to view information and recommended outfits shall take no longer than 5 seconds.

3.2.2 Safety Requirements

- The system must not halt or lag, especially during the update time and must not go down under high traffic. In order to ensure safety of the server, it is suggested that it is hosted on two computers - one kept as a backup.

3.2.3 Security Requirements

- It must be ensured that only the authorized admins, with valid user credentials, have access to the data of the users in order to ensure user privacy.
- The system will use databases from authentic sources and fashion stores.

3.2.4 User Interface

- The UI/UX flow needs to be intuitive as well as modern.

3.2.5 Error Handling

- The system prevents data loss by carefully handling all expected and non-expected errors.

3.3 External Interfaces

3.3.1 User Interfaces

Customer Interface

- Homepage

  This interface would be visible to all the users and would lead to multiple other interfaces such as Login, Clothing categories, User profile, etc. This is shown in Figure 3.1.

- Registration

  This page allows a new user to create an account by filling the mandatory fields of 'User Name', 'Email', 'Password' and 'Confirmed Password'. In case of valid details, user will be able to login, else they are redirected to the same sign-up page. This is shown in Figure 3.2.

- Login

  This interface enables user to log into the system using valid credentials, and redirects him/her to the homepage if the credentials are validated, otherwise an error message is displayed. Login interface requires 'User Name' and 'Password' as the mandatory fields. This is shown in Figure 3.3.

- Profile

  A user would be able to see this interface if they have created an account and are logged into the system. This interface would enable them to view their account details and their previously searched/recommended outfit statistics. This is shown in Figure 3.4.

- Product Display

  A user would be able to see different product categories along with the details of each product available in the stock. This is shown in Figure 3.5, Figure 3.6, and Figure 3.7.

- Upload Image

  A user would be able to upload image of his/her clothing item, in order to search visually similar or complementary items. This is shown in Figure 3.8 and Figure 3.9.

- Cart

  This interface enables a user to view their selected recommended outfits and would show them the third-party referral links to each of their chosen outfits. This is shown in Figure 3.10.

- Our Brands

  This interface would display the vendors and the third-party brands with whom we will partner up with. A user would be able to retrieve products based on any specific store from these brands. This is shown in Figure 3.11.

System Admin Interface

- Login

  Using the login interface, the system admin would log into the system using his/her valid credentials and would be redirected to the system admin homepage. This is shown in Figure 3.3.

- Homepage

  This interface enables the system admin to get redirected to multiple other navigation pages such as User Details, and Vendor details. This is shown in Figure 3.1.

- User Details

  This interface contains user details such as their personal information, their cart details and their preferred trend statistics. This is shown in Figure 3.12 and Figure 3.13.

- Vendor Details

  This interface allows the system admin to view details of the vendor such as the trends of their most recommended outfits, new additions to their outfit database, top users visiting the respective vendor page (using the referral link). This is shown in Figure 3.14 and Figure 3.15.

GUI Mockups

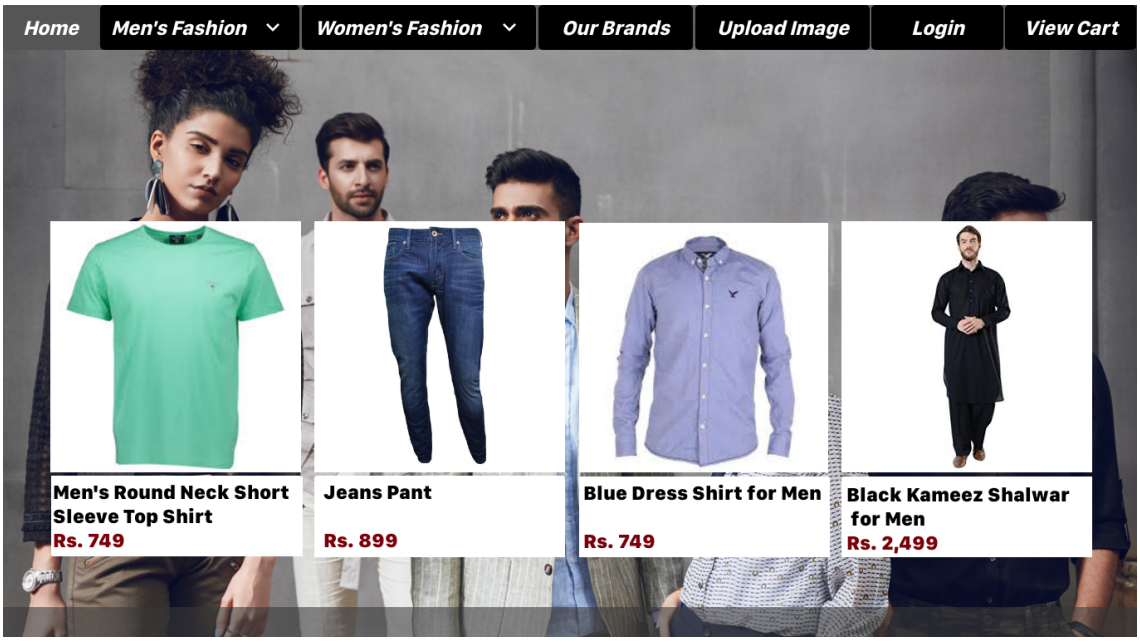

Figure 3.1: Home Screen

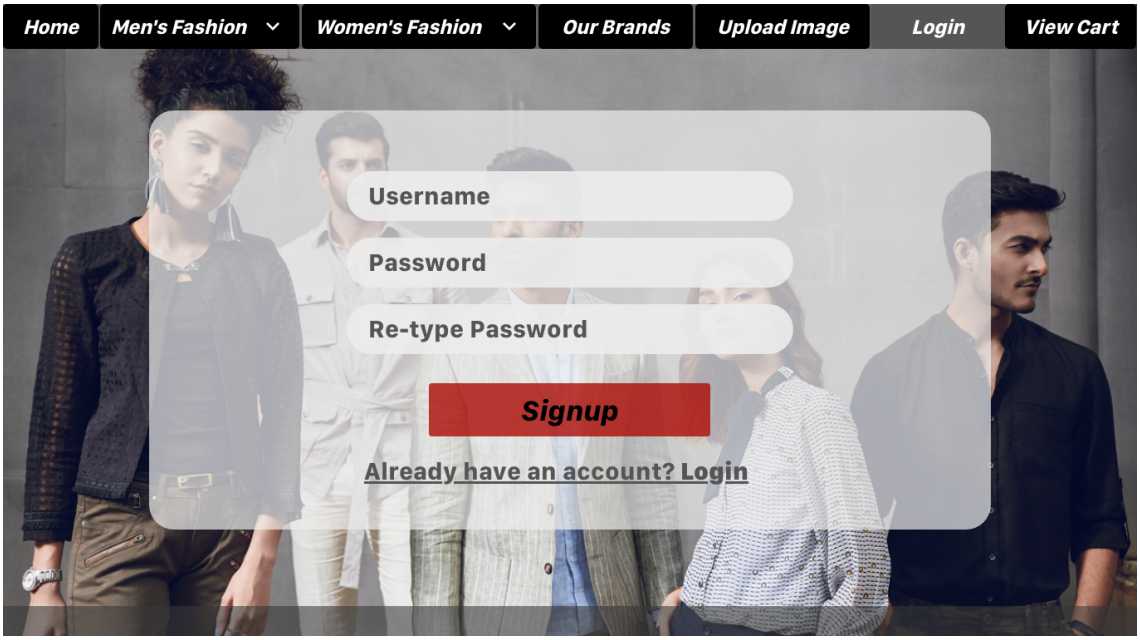

Figure 3.2: Sign-up Screen

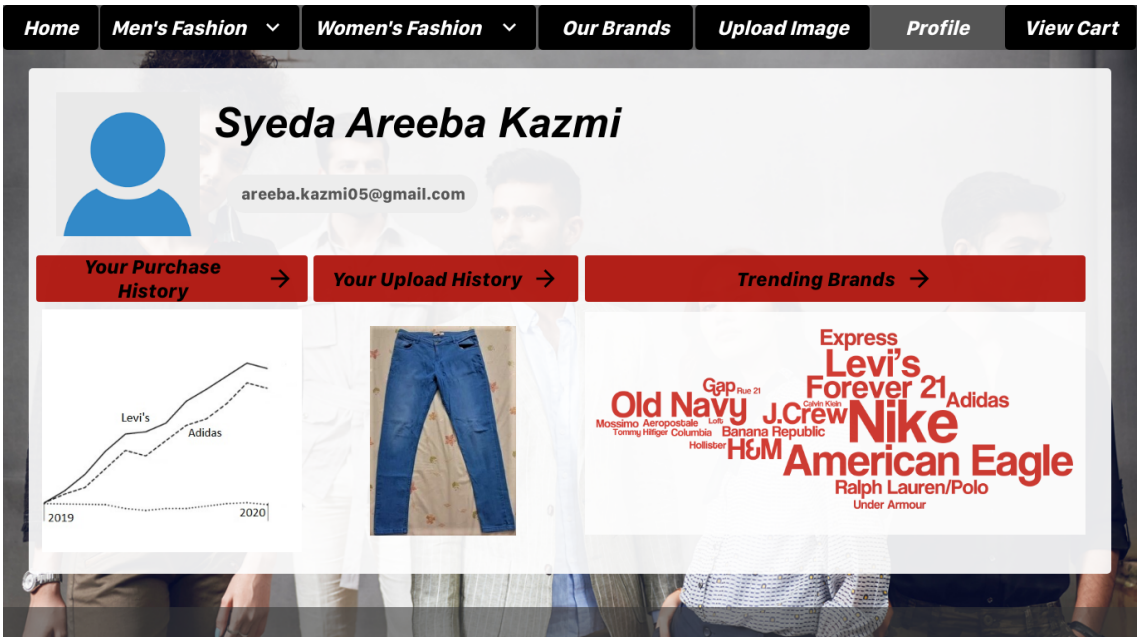

Figure 3.4: User Profile Screen

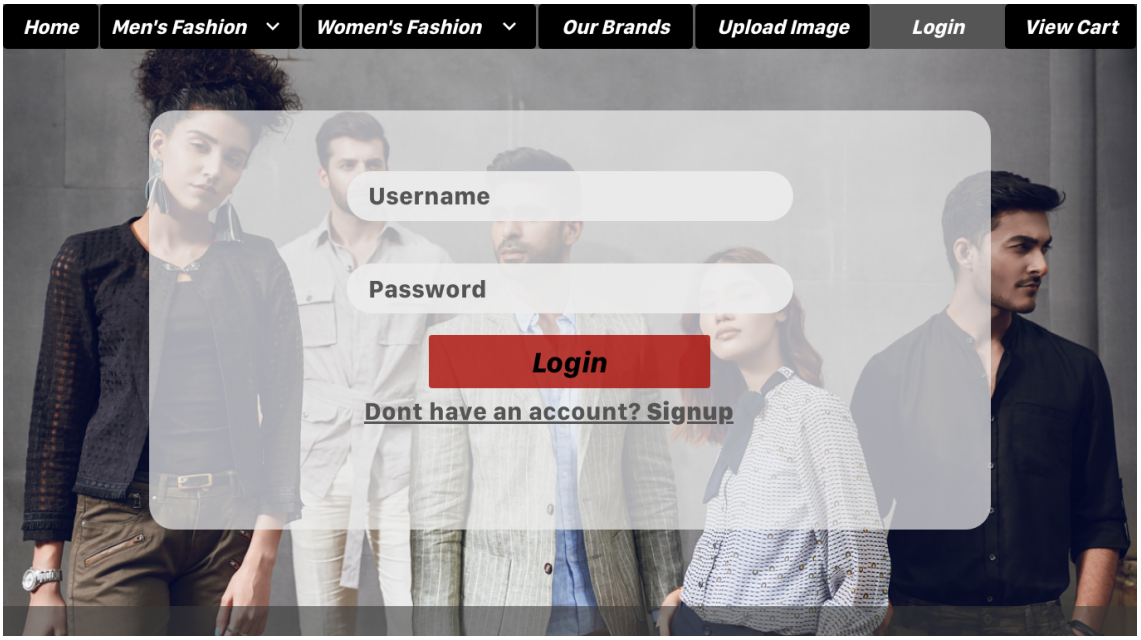

Figure 3.3: Login Screen

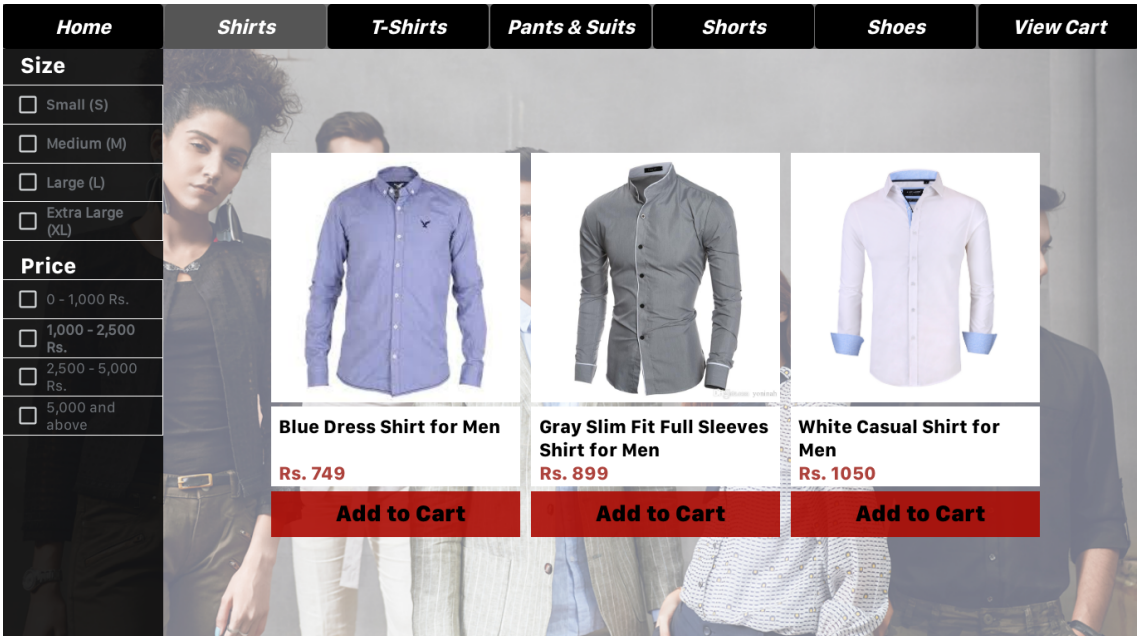

Figure 3.5: Shirt Category Screen

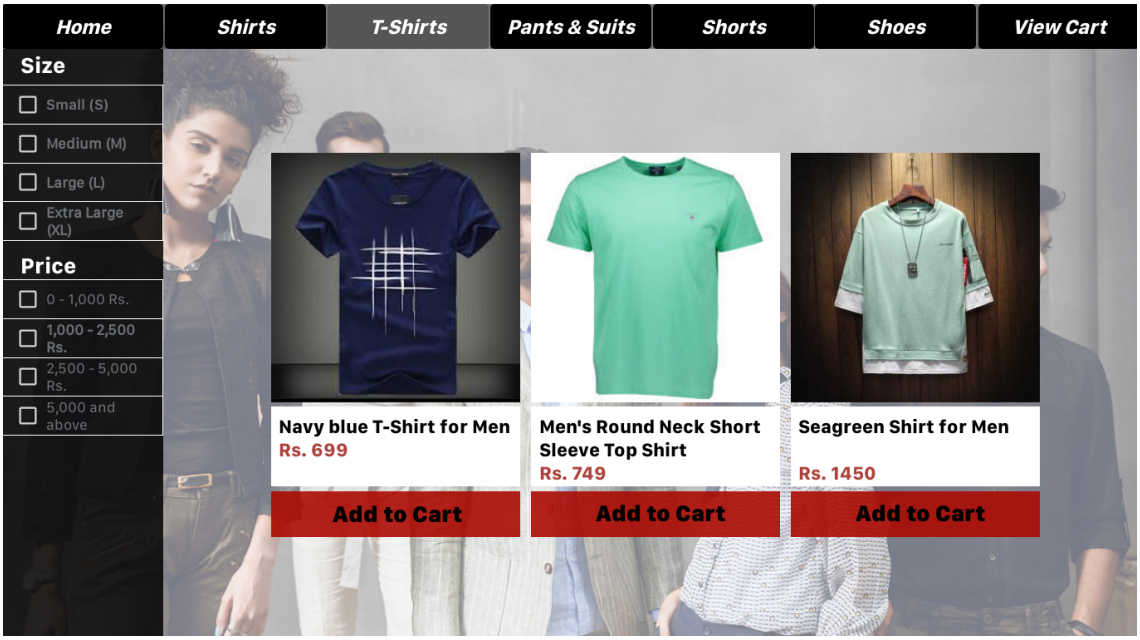

Figure 3.6: T-Shirt Category Screen

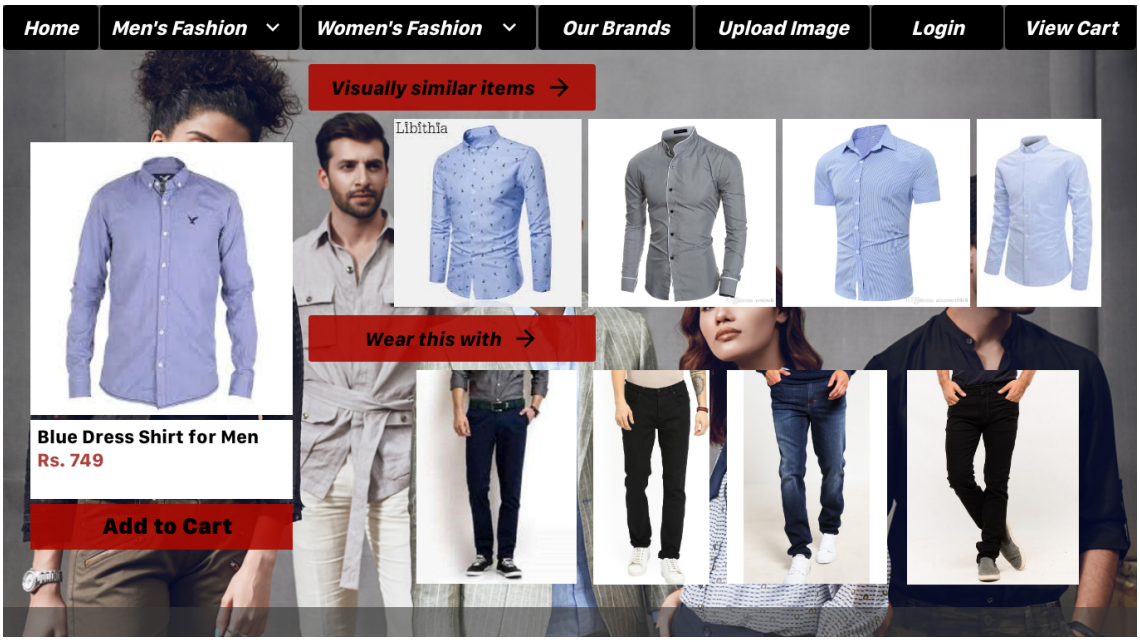

Figure 3.7: Recommendation Screen 1.0

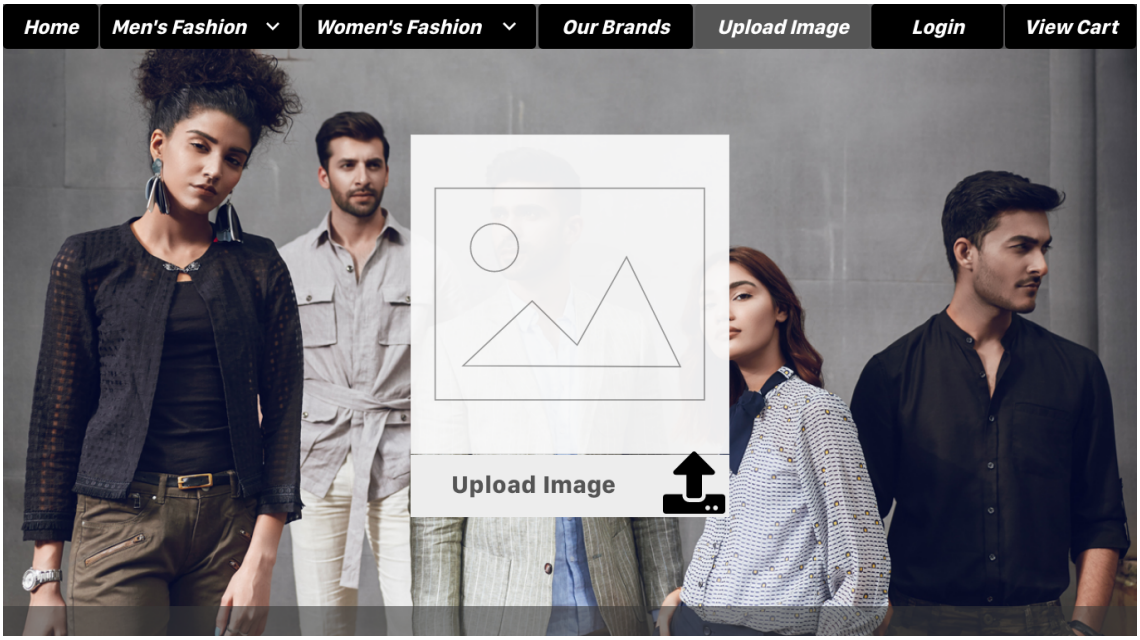

Figure 3.8: Upload Image Screen

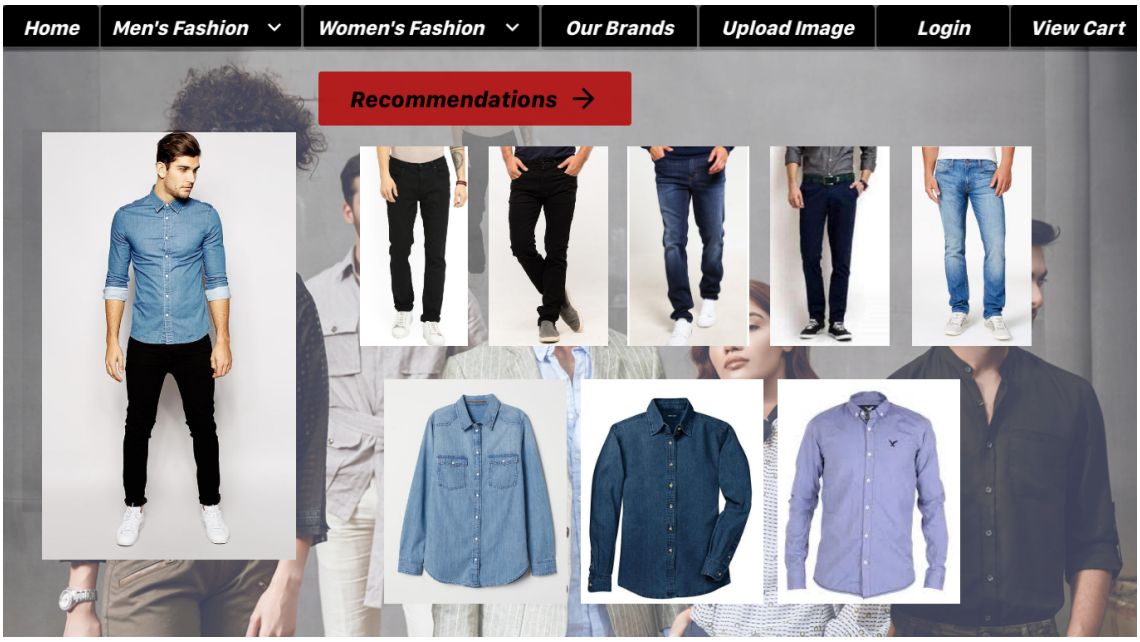

Figure 3.9: Recommendation Screen 2.0

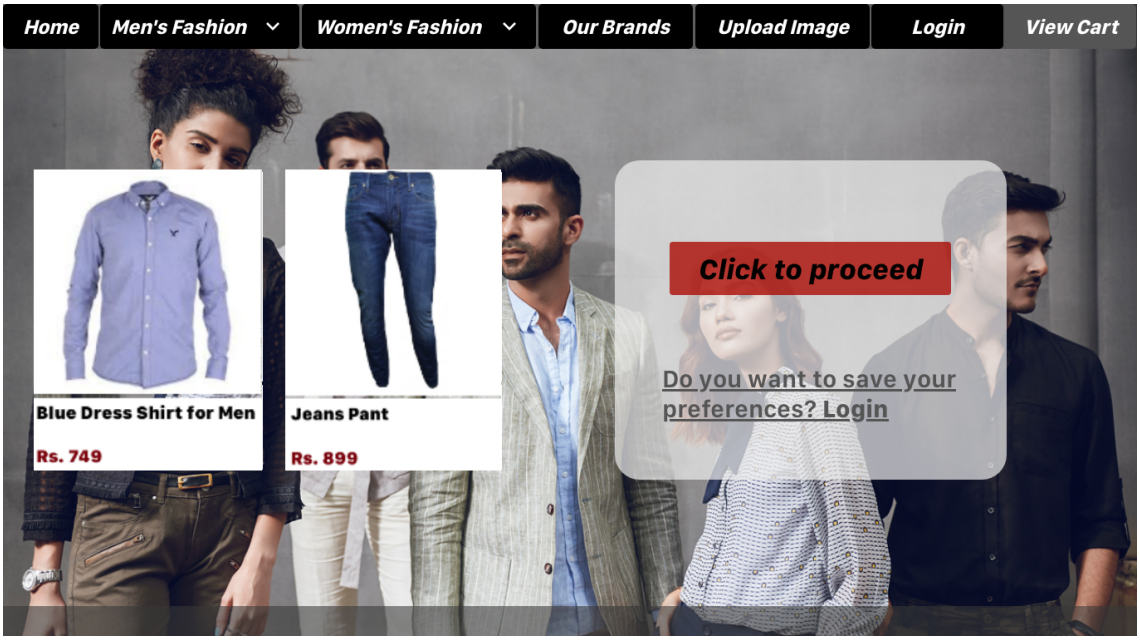

Figure 3.10: View Your Cart Screen

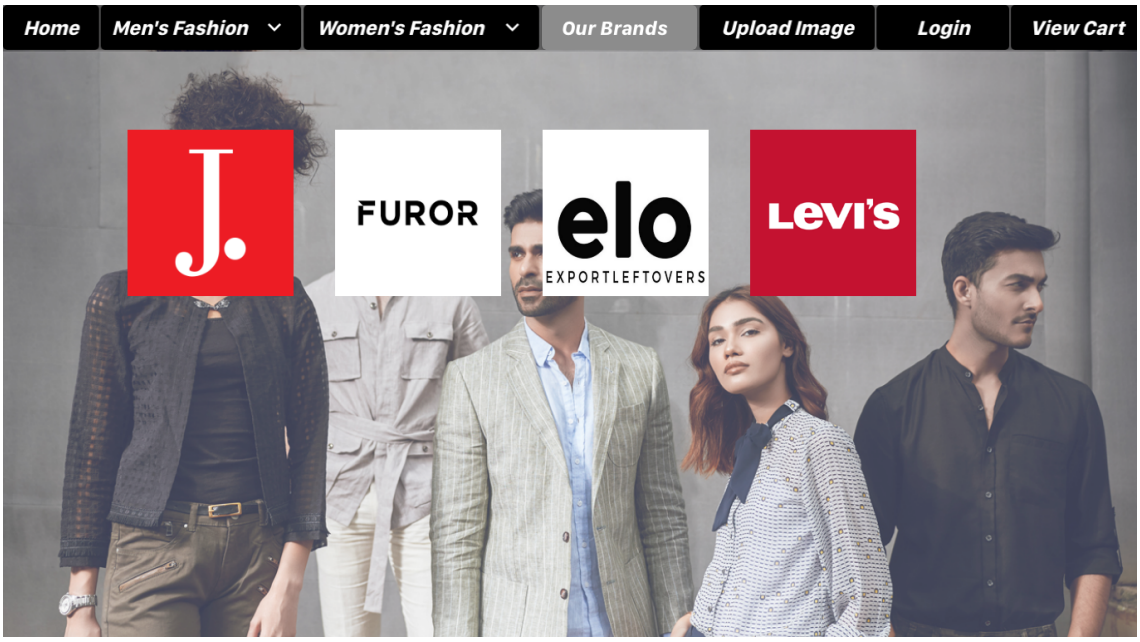

Figure 3.11: Partner Brands Screen

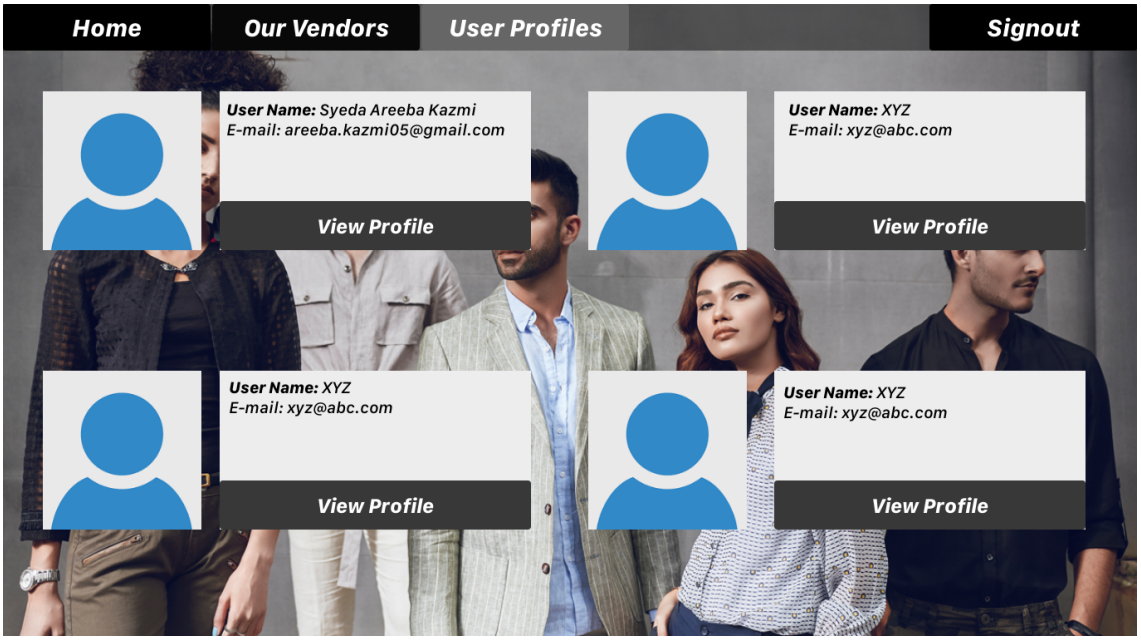

Figure 3.12: Admin-User Profiles Screen 1

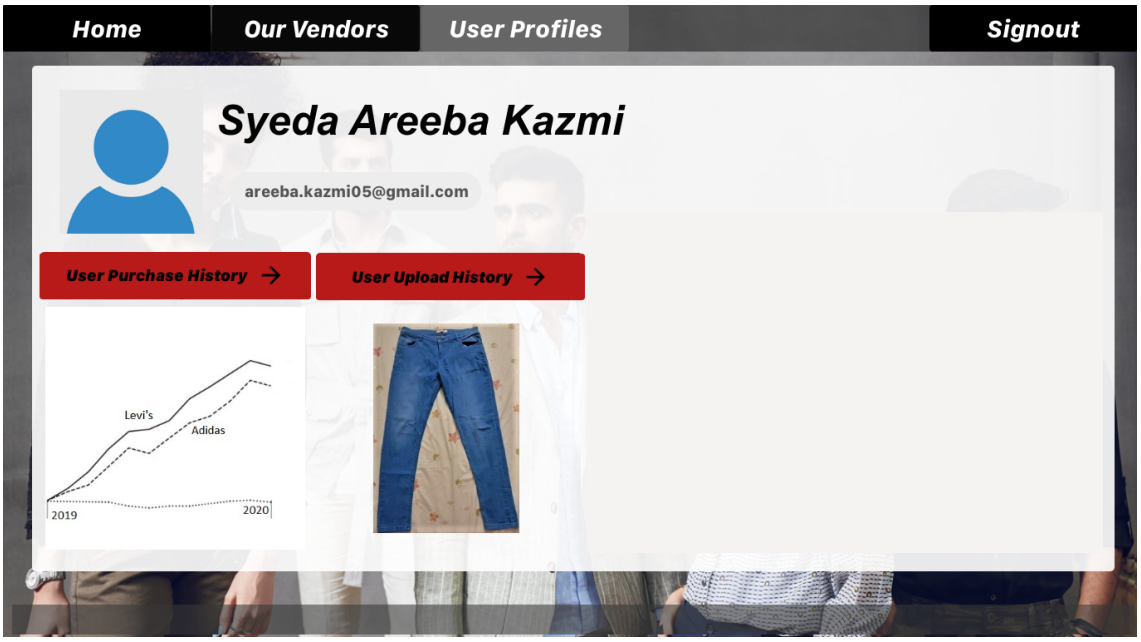

Figure 3.13: Admin-User Profiles Screen 1

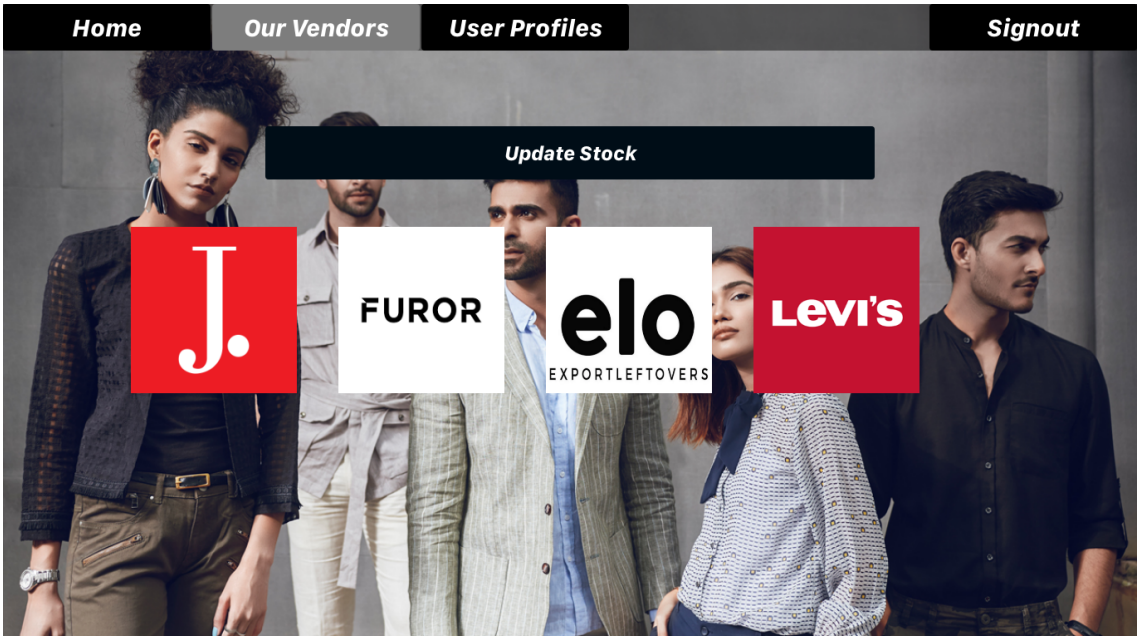

Figure 3.14: Admin Vendors Screen 1

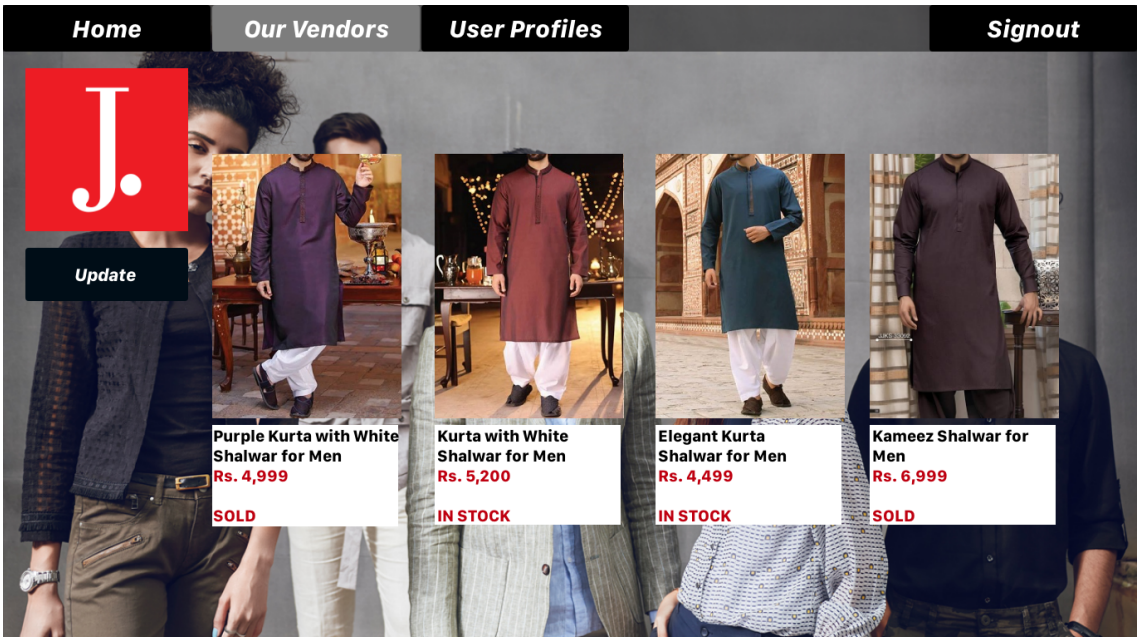

Figure 3.15: Admin Vendors Screen 2

3.3.2 Application Program Interface (API)

The system will be built using the Django framework at the server-end, and React.JS at the client-end (explained in detail in section 3.6). We aim to use REST (Representational State Transfer) APIs, served through the Django REST framework, for all client-server communication since RESTful services as an architectural style would lead to our application being lightweight and scalable. However, since graph-based communication channels are gaining more popularity and adaptation rates in internet technologies these days, and for reasons summarized in Table 2.3 ahead, our eventual goal is to establish an API based entirely off of GraphQL. So, for purposes of this web-application, our API will be written primarily as a RESTful service, but will be exposed to the client through a schema-first approach using a GraphQL wrapper. Figure 3.16 summarizes this.

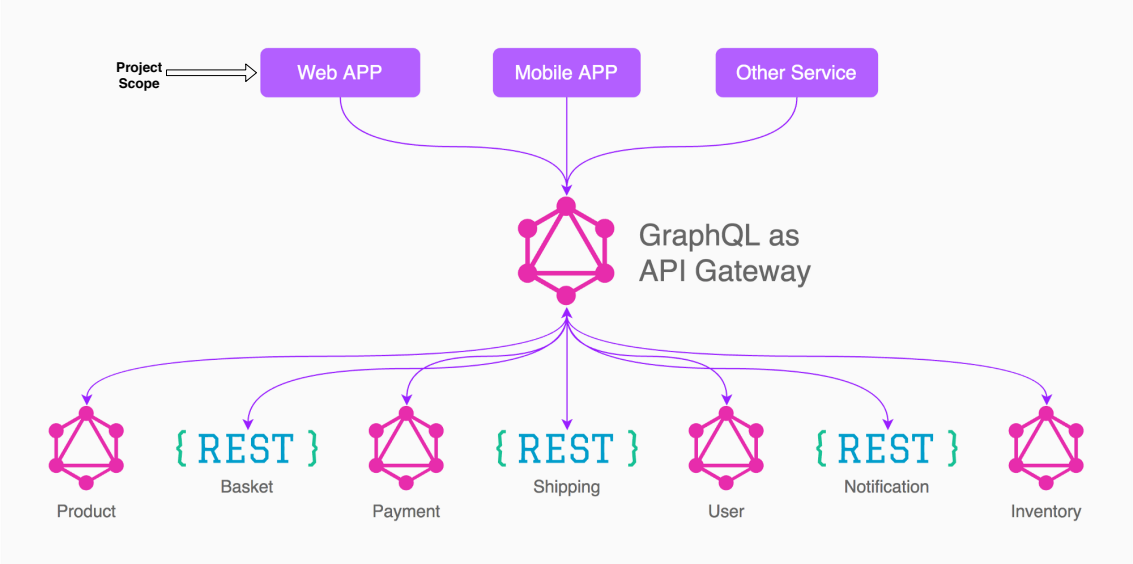

Figure 3.16: API Diagram

3.3.3 Communication Interfaces

Based on the project requirements and keeping the communication channels in mind, the chosen technology stack for the client-end is: TypeScript, React, and React Apollo (for interacting with the GraphQL API), and for the server-end is: Python, Django, and Graphene Django (for creating the GraphQL API). Table 2.1, Table 2.2,

and Table 2.3 offer a detailed comparison between why these communication interfaces were picked over some of their major competitors.

3.4 Use Cases

The two main users of the system are customers and the admin. Figure 3.17 summarizes their system use cases.

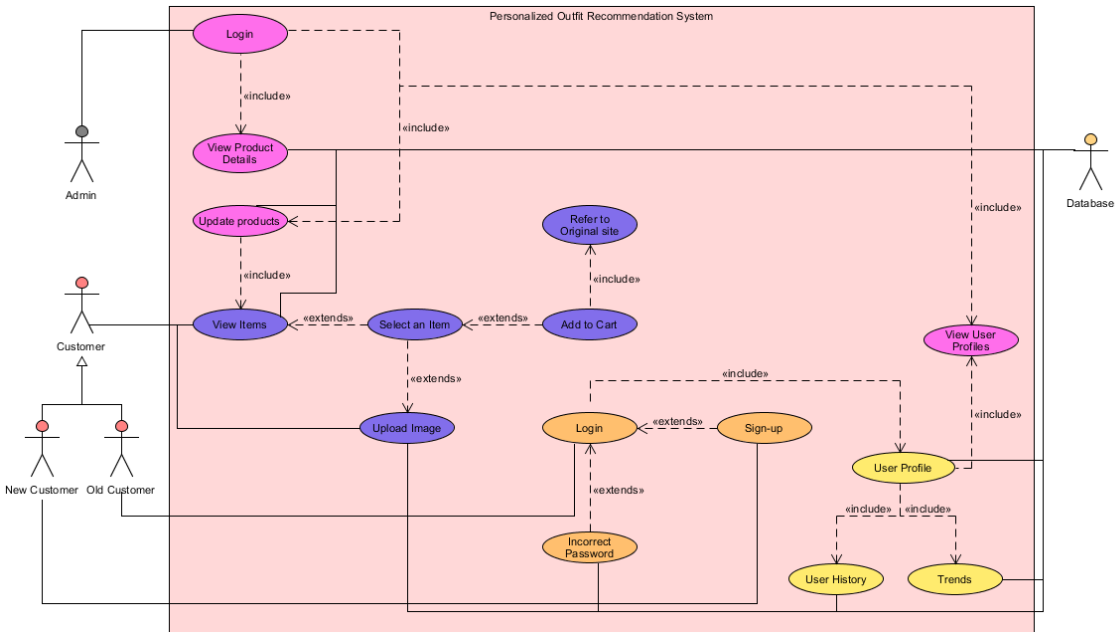

Figure 3.17: Use-Case Diagram

3.4.1 User Login

The following table provides information regarding the process when user attempts to log into the system, describing the conditions and conclusion if the attempt succeeds.

Use-Case Name:	Login
Use-Case ID:	PORS-01
Priority:	High
Primary Business Actor:	System Admin
Other Participating Actors:	Customer
Description:	System admin needs to login in order to view customer, product details and update the stock. Whereas, the customer needs to login to view the referred sites and save his profile
Pre-condition:	User has registered previously on the app and has valid email and password.
Trigger:	This use case is initiated when the admin wants to navigate the app or customer tries to view the referred site.
Typical course of events:	**Actor Action:** Actor enters valid username and password. **System Action:** Checks if the given username is registered in database and the given password is valid.
Alternate courses:	If the credentials are invalid then an error message is displayed.
Conclusion:	User successfully logs in using valid credentials.
Post-condition:	User login is logged in the database.

3.4.2 View Product Details

The following table provides information regarding the process when user attempts to view product details, describing the conditions and conclusion if the attempt succeeds.

Use-Case Name:	View Product Details
Use-Case ID:	PORS-02
Priority:	High
Primary Business Actor:	System Admin
Other Participating Actors:	Customer
Description:	System Admin will view the product details to be able to update the stock. Whereas the customer will be able to see the product with details in order to select the items of their choice.
Pre-condition:	Logged in as Admin or Customer.
Trigger:	This use case is triggered when the user clicks a product.
Typical course of events:	Appropriate product details will be displayed to the user.
Alternate courses:	None
Conclusion:	The user is able to see the product detail page.
Post-condition:	User search is logged in the database.

3.4.3 Update Stock

The following table provides information regarding the process when user attempts to update product stock, describing the conditions and conclusion if the attempt succeeds.

Use-Case Name:	Update Stock
Use-Case ID:	PORS-03
Priority:	High
Primary Business Actor:	System Admin
Other Participating Actors:	None
Description:	System Admin will be able to update new stock and product details in case a product is sold out.
Pre-condition:	Logged in as Admin.
Trigger:	This use case will be triggered when the system admin clicks update stock button.
Typical course of events:	**System Action:** System will check all the products and update the sold out and new products accordingly.
Alternate courses:	None
Conclusion:	The stock is successfully updated.
Post-condition:	Customers are able to view the updated stock.

3.4.4 Upload an Image

The following table provides information regarding the process when user attempts to upload an image on the web application, describing the conditions and conclusion if the attempt succeeds.

Use-Case Name:	Upload an Image
Use-Case ID:	PORS-04
Priority:	High
Primary Business Actor:	Customer
Other Participating Actors:	None
Description:	Application will allow customers to upload an image of their piece of clothing in order to find a similar or complementary items.
Pre-condition:	None
Trigger:	This use case will be triggered when the user will click upload image option.
Typical course of events:	**System Action:** System will check if the uploaded photo is of a piece of clothing
Alternate courses:	If the uploaded image is not recognizable, a suitable error message is displayed.
Conclusion:	Customer's valid picture will be uploaded on the app.
Post-condition:	Customer will be able to see the recommendation according to the uploaded picture.

3.4.5 Add to Cart

The following table provides information regarding the process when user attempts to add products to the cart for shopping, describing the conditions and conclusion if the attempt succeeds.

Use-Case Name:	Add to Cart
Use-Case ID:	PORS-05
Priority:	High
Primary Business Actor:	Customer
Other Participating Actors:	None
Description:	Items selected by the customer will be added to the cart in order to proceed to the original referral site.
Pre-condition:	None
Trigger:	This use case will be triggered when the customer will select an item and click the option 'add to cart'.
Typical course of events:	System will check if the product is available to be added to the cart.
Alternate courses:	None
Conclusion:	Selected item will be added to the customer's cart.
Post-condition:	Valid referral link to the item is updated.

3.4.6 Refer to Original Sites

The following table provides information regarding the process when user attempts to view original websites from where the product has been taken from, describing the conditions and conclusion if the attempt succeeds.

Use-Case Name:	Refer to Original Sites
Use-Case ID:	PORS-06
Priority:	Medium
Primary Business Actor:	Customer
Other Participating Actors:	None
Description:	Once the customer has selected items of his choice, those will be added to the cart and then he would be redirected to the original product site.
Pre-condition:	Items added to the cart
Trigger:	This use case will be triggered when a customer clicks on 'proceed to original site' option, available in his cart.
Typical course of events:	**System Action:** System will direct the customer to the original sites of the items available in customer's cart
Alternate courses:	None
Conclusion:	Customer directed to the original sites of the items he has chosen.
Post-condition:	Reference point is logged in the dataset.

3.4.7 Sign-up

The following table provides information regarding the process when user attempts to sign up a new account, describing the conditions and conclusion if the attempt succeeds.

Use-Case Name:	Sign-up
Use-Case ID:	PORS-07
Priority:	High
Primary Business Actor:	Admin
Other Participating Actors:	Customer
Description:	Admin and Customers will have to sign-up and register on the app in order to login and proceed.
Pre-condition:	Valid information by the user.
Trigger:	This use case will be triggered when a new user wants to register on the application.
Typical course of events:	**Actor Action:** Providing valid details. **System Action:** Checking if the details provided are available in the database.
Alternate courses:	If the details provided are incomplete or already registered then appropriate error message must be displayed.
Conclusion:	User is able to sign-up successfully.
Post-condition:	New user is added to the database.

3.4.8 Incorrect Password

The following table provides information regarding the process when user attempts to enter log into the system using an incorrect password, describing the conditions and conclusion if the attempt succeeds.

Use-Case Name:	Incorrect Password
Use-Case ID:	PORS-08
Priority:	High
Primary Business Actor:	Admin, Customer
Other Participating Actors:	None
Description:	While logging-in on the app, the user must enter a valid email and password.
Pre-condition:	Registered on the app.
Trigger:	This use case will be triggered when a user tried to login with incorrect password or username.
Typical course of events:	**System Action:** System checks if the given password or user-name is registered in the database.
Alternate courses:	System displays an error message about incorrect credentials.
Conclusion:	User unable to login to the app.
Post-condition:	Incorrect login attempt is logged in the database.

3.4.9 User Profile

The following table provides information regarding the process when user attempts to view his/her profile, describing the conditions and conclusion if the attempt succeeds.

Use-Case Name:	User Profile
Use-Case ID:	PORS-09
Priority:	Medium
Primary Business Actor:	Customer
Other Participating Actors:	System Admin
Description:	All registered users will have user profiles displaying their details, purchase history and uploaded image history.
Pre-condition:	User must be logged in.
Trigger:	This use case will be triggered when the admin wants to look at a particular user's profile or the customer himself clicks at his profile.
Typical course of events:	System will display the updated user profile to the user.
Alternate courses:	None
Conclusion:	User profile displayed.
Post-condition:	None

3.4.10 User History

The following table provides information regarding the process when user attempts to view user shopping history, describing the conditions and conclusion if the attempt succeeds.

Use-Case Name:	User History
Use-Case ID:	PORS-10
Priority:	Medium
Primary Business Actor:	Customer
Other Participating Actors:	System Admin
Description:	All registered users will have their user history comprising of purchase history and uploaded image history.
Pre-condition:	1. User is registered in the database. 2. User has either purchased or uploaded an image before.
Trigger:	This use case will be triggered when a user purchases an item or uploads any image.
Typical course of events:	System will update the user history when the user will purchase or upload an item.
Alternate courses:	None
Conclusion:	User history is updated at every purchase or upload, given that the user is logged in.
Post-condition:	None

3.4.11 View Trends

The following table provides information regarding the process when user attempts to view details of the latest fashion trends, describing the conditions and conclusion if the attempt succeeds.

Use-Case Name:	View Trends
Use-Case ID:	PORS-11
Priority:	Medium
Primary Business Actor:	Customer
Other Participating Actors:	None
Description:	Different statistics will be displayed indicating the sales or ratings of each brand in last 30 days.
Pre-condition:	users have either purchased or rated an item from any brand.
Trigger:	This use case will be triggered when the user clicks trend option.
Typical course of events:	System will update the trend at each purchase or rating by the user.
Alternate courses:	None
Conclusion:	User is able to see the most updated trends.
Post-condition:	Catalog matrix is updated in the database.

3.5 Datasets

This section describes the specific dataset which will be used to build our system. An appropriate snapshot of the dataset is also included in in the description below which further signifies the relevance of the used dataset. Futher details regarding the daatset are provided in the appendix below.

For purposes of the image classification pipeline for our recommendation engine, we have selected DeepFashion [16] as our preliminary dataset. The dataset consists over 800,000 diverse fashion images ranging from well-posed shop images to unconstrained consumer photos. What makes this dataset suitable towards our system is the fact that it is annotated with rich information of clothing items. Each image in this dataset is labelled with 50 categories, 1,000 descriptive attributes, bounding box and clothing landmarks, and this would kick-start the training of our classifier. A category-wise subset of the dataset is depicted in Figure 3.18.

It is important to clarify here that the DeepFashion dataset consists of product images from western stores and outlets, so the classifier would not generalize efficiently to products in the local eastern domain, which is a requirement of our

system. Moreover, there are limitations in the amount of products and datasets we can scrape from eastern fashion stores such as J., Export Leftovers, Zellbury, Daraz, etc. If we were to use these as the only data for training our classifier, the model would indubitably suffer from underfitting. In order to maintain a good fit, we will use the concept of Transfer Learning to re-train the classifier trained on DeepFashion on our local scraped dataset from the aforementioned local fashion stores.

Figure 3.18: The DeepFashion Dataset

3.6 System Diagram

Figure 3.19 gives an overview of different modules of our system, highlighting some of the main functionalities that would be play an efficient role for the smooth working of our project. The main modules of our system are provided as follows:

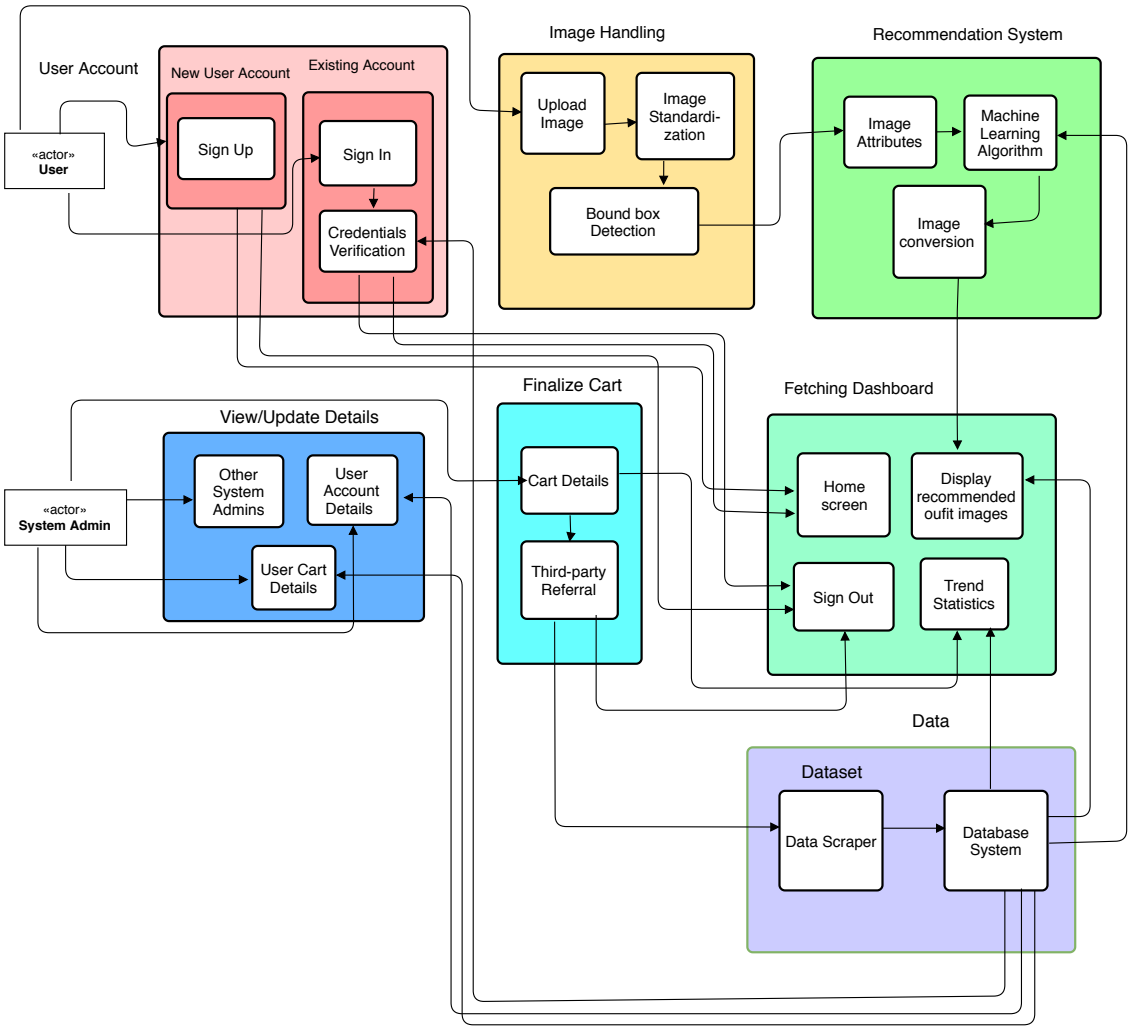

Figure 3.19: Module-wise System Diagram

Figure 3.20 gives an overview of the chosen architecture of our system that is different technologies for the backend, frontend and API which will be sued for the development of this project are provided as follows:

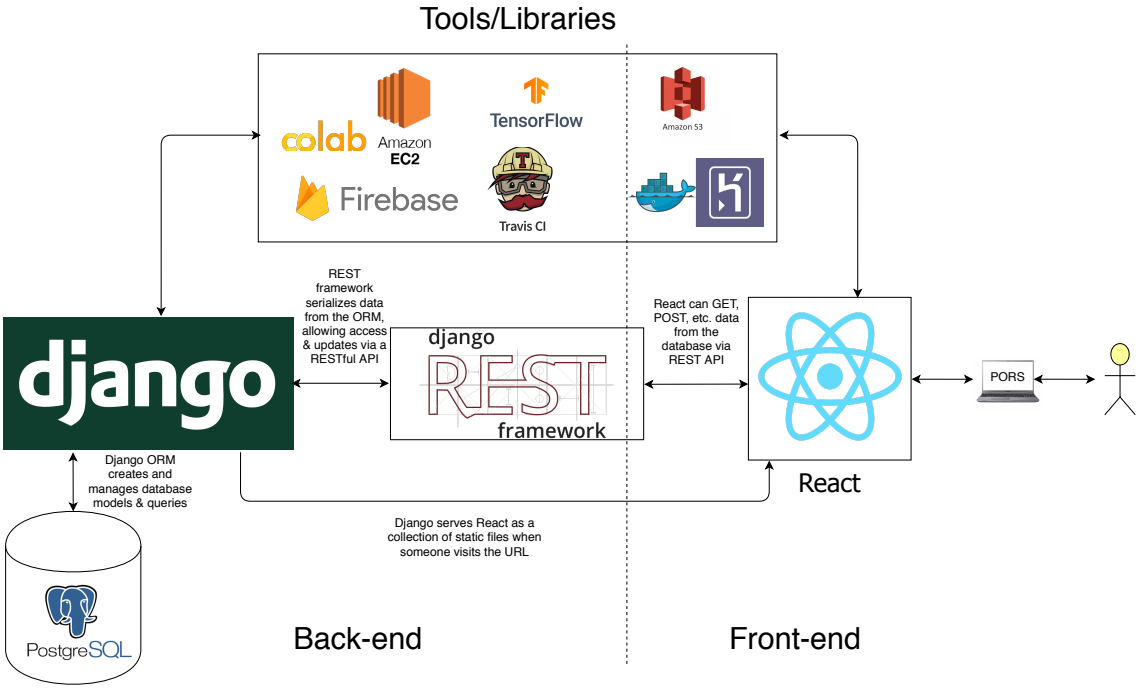

Figure 3.20: Technology-wise System Diagram

3.7 Data Flow Diagram

Rudimentary data flow diagrams for the system have also been constructed, given below in Figure 3.21 and Figure 3.22:

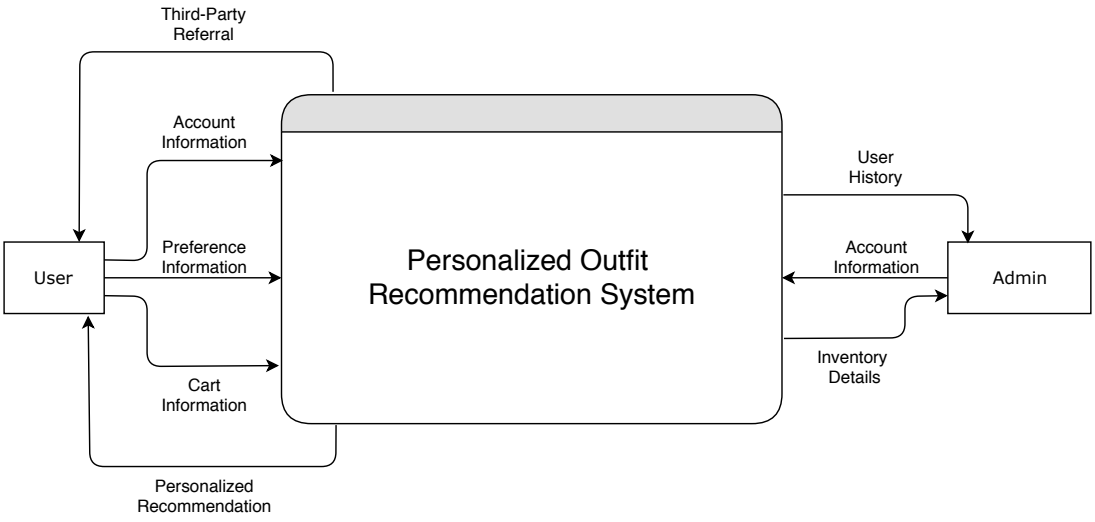

Figure 3.21: Context Level DFD

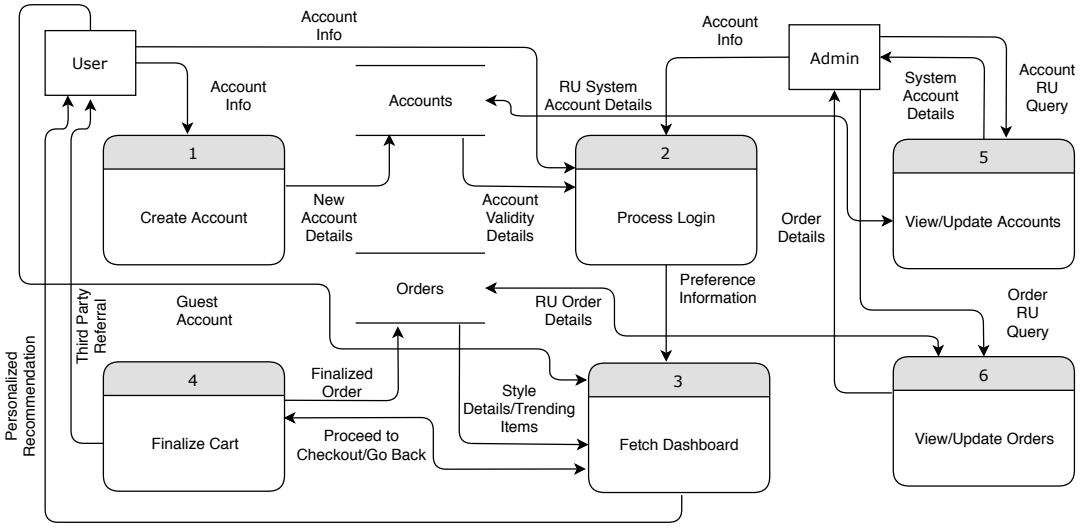

Figure 3.22: 0-Level DFD

3.8 Association Matrices

	User	System Admin
User	Individual	All
User ID	R	CRUD
Name	RU	CRUD
Email ID	CRU	RD
Password	RU	RD
Sign in/Sign up/ Sign out	Individual	All
Account ID	R	RD
Date	R	RD
Verify Credentials	Individual	All
Account Valid	R	CRUD
Logged In	R	CRUD
Date	R	CRUD
Fetch Dashboard	Individual	All
Page ID	R	CRUD
Image Details	R	CRUD
Finalize Cart	Individual	All
Cart ID	R	CRUD
Date	R	CRUD
Item No.	RU	CRUD
Third-party Referral	R	CRUD
Recommendation Details	Individual	All
Recommendation ID	R	CRUD
User ID	R	CRUD
Date	R	CRUD
Image	Individual	All
Image ID	R	CRUD
Category	R	CRUD
Recommendation ID	R	CRUD

Table 3.1: CRUD: Data to Location Matrix

	Account	Credentials	Dashboard	Cart	Image
User					5
User ID	R	R	R	R	R
Name	CRU	RU			
Email	CRU	R			R
Password	CRU	R			
Sign in/up/out					
Account ID	R	R			
Date	R	R			
Verify Credentials					
Account Valid		RU	R		
Logged In		RU	R		
Date		R	R		
Fetch Dashboard					
Page ID			R	R	
Image Details			RUD	RU	
Finalize Cart					
Cart ID			R	R	
Date			R	RUD	
Item No.			R	RUD	
Third-party Referral			R	RUD	
Recommendation Details					
Recommendation ID			R	RU	
User ID			R	RUD	
Date			R	RUD	
Image					
Image ID			R	R	R
Category			RUD	RUD	R
Image Attribute			RUD		R
Recommendation ID			R	R	R

Table 3.2: CRUD: Data to Process Matrix

	User	System Admin
Process Account	*	
Process Verify Credentials		*
Process Dashboard		*
Process Cart	*	
Process Image Handling		*

Table 3.3: CRUD: Process to Location Matrix

4. Software Design Specification (SDS)

This chapter provides important artifacts related to design of our project. The designs provided in this chapter attempt to provide a comprehensive study of the model of our software design including the entity relationship diagram which provides a comprehensive relationship model between different models (or rather, tables) of our database system. The state diagram and the sequence diagram also describe the flow of the software about how the application attempts to proceed.

4.1 Data Design

This section presents the structure of our database that caters to persistent data storage in our project. The structure is shown as a normalized data model for relational databases. It clearly shows entities, attributes, relationships with their cardinalities, and primary and foreign keys. We have used DB designer to build our data model.

The following figure shows an extensive *entity relationship model* between different tables of the system, where each entity is a different relational table in the database, having one - to - one or one - to - many relationships. The many - to - many relations are resolved by introducing a separate relationship resolver entity.

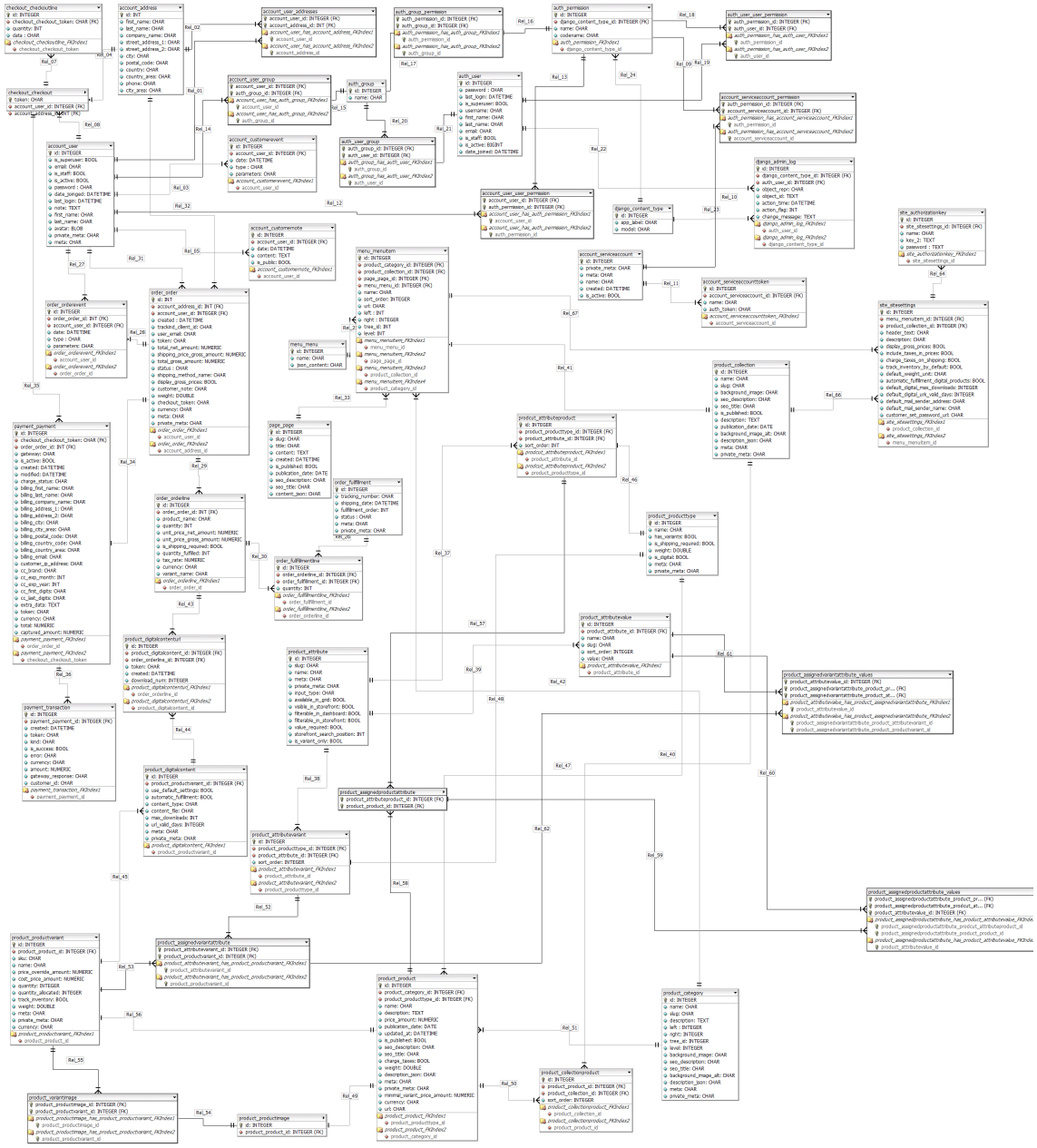

Figure 4.1: Entity Relationship Diagram

4.2 State Diagram

The state diagram provides an understanding of the different stages that the application can be in depending upon the activity of the user/customer. The first diagram shows the process of category based shopping while the second diagram shows the process of image upload.

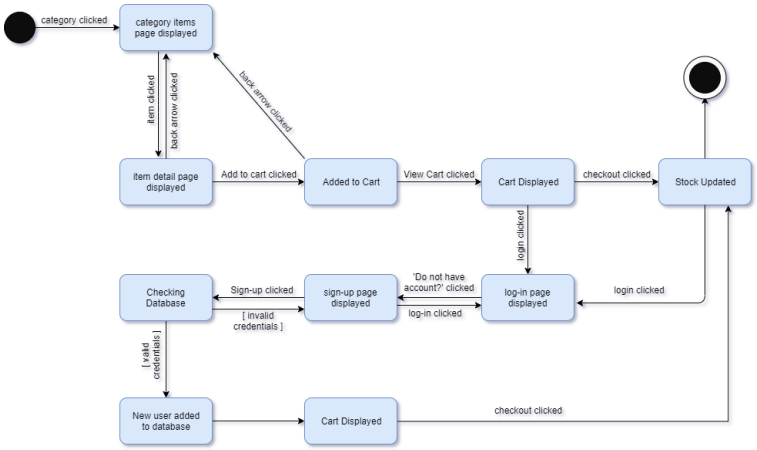

Figure 4.2: State Diagram 1

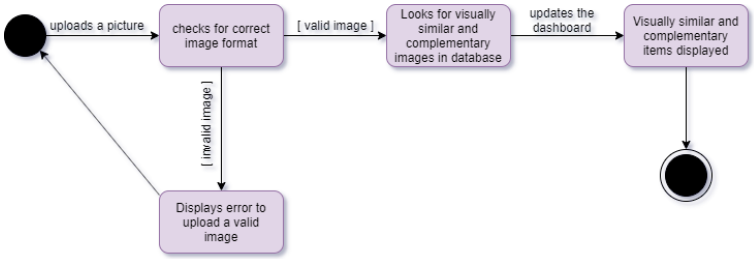

Figure 4.3: State Diagram 2

4.3 Sequence Diagram

The sequence diagram attempts to provide a holistic picture of different stages of a customer using our application. Each action also shows the path followed by the application when the action of the user succeeds (in green) and the path that the user is redirected to when that action fails (in red).

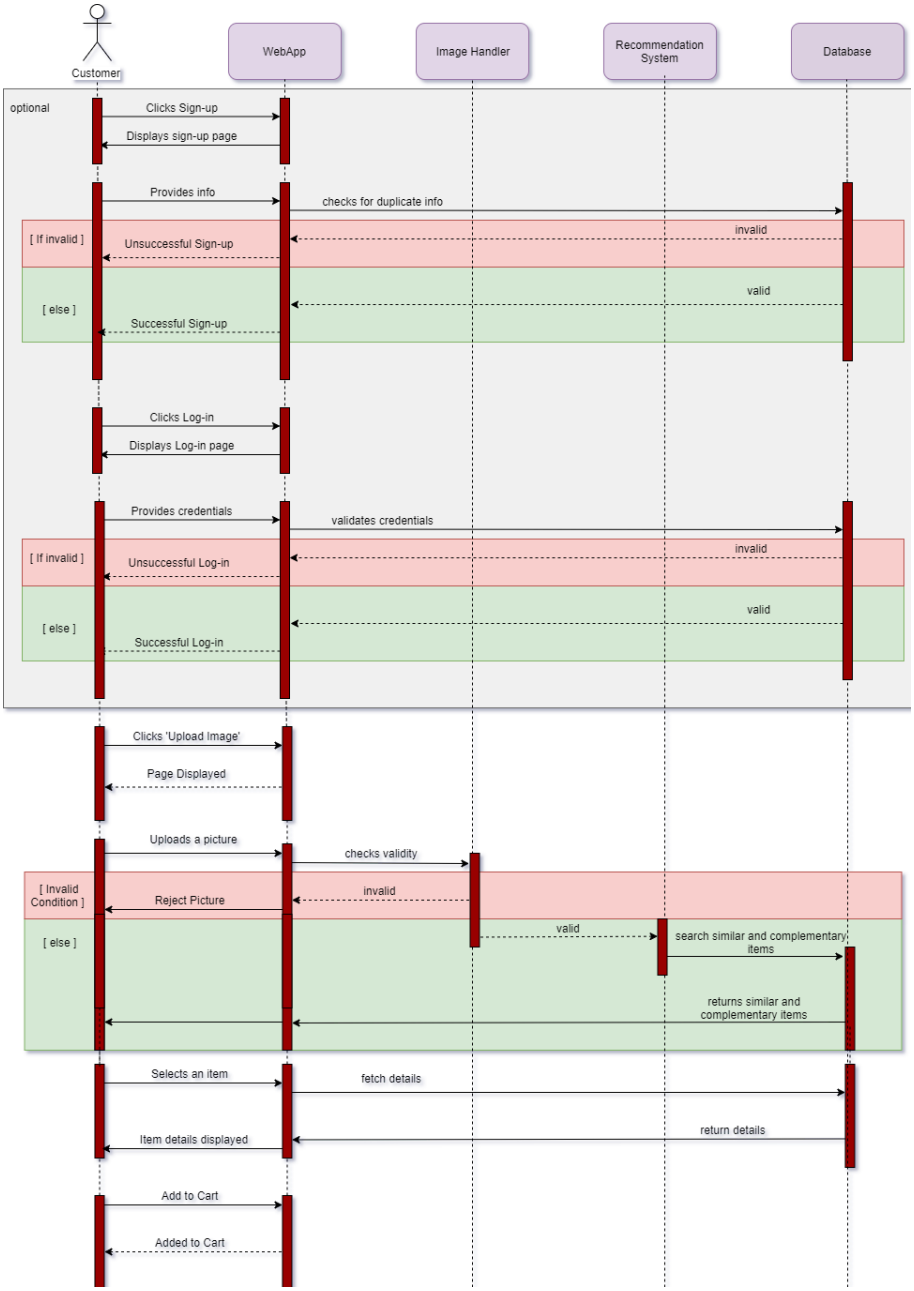

Figure 4.4: Sequence Diagram - Customer

5. Experiments and Results

The deep learning model that powers our recommendation systems visua'l search was iterated upon and consistently improved during the course of our Kaavish. We also made a concerted effort to ensure that our AI model was thoroughly tested and benchmarked at every experiment.

Experimentation- stages of evolution of the AI model		
Experiment. no	**Number of Categories Trained On**	**Transfer Learning**
1	20	Yes
2	26	Yes
3	50	No

Table 5.1: Experimentation- stages of the AI model

Due to limitations in computing power, we initially trained the model on only the first 20 categories of the classification benchmark of the DeepFashion dataset. These categories correspond to the "top" clothing type (i.e. clothes worn on the top part of the body like shirts and coats, as opposed to items worn on the bottom like jeans).

Once the viability of the model was proved in Kaavish I, we experimented by training our model on 26 categories that were selected based on their relevance to mens clothing, since that is the focus of our platform. Categories containing a large proportion of images of men's clothing were included while those containing a majority of women's clothing were excluded.

The 26 categories included were: Anorak, Blazer, Bomber, Button-Down, Cardigan, Flannel, Henley, Hoodie, Jacket, Jersey, Parka, Peacoat, Sweater, Tank, Tee, Top, Turtleneck, Chinos, Jeans, Joggers, Shorts, Sweatpants, Sweat-shorts, Trunks, Coat and Robe. Other clothing categories such as blouse and skirt, which almost exclusively contained images of women, were excluded.

The motivation behind this experiment was to solve the problem of the lack of diversity in the subset of the DeepFashion dataset used to train the first model, which only contained items of the "top" clothing type. By including only those categories we deemed to be relevant to a men's fashion e-commerce store, we aimed to make our model more accurate while still minimizing computational cost.

In the course of this experiment, we were provided access to Habib University's new High Performance Computer, which contains an Nvidia Titan X GPU. With this significant increase in computational power at our disposal, we chose to train our model on all 50 categories of the DeepFashion dataset and so, no longer needed to complete the second experiment which we initially planned to carry out for Transfer learning. So its results are not available.

Although many of the newly included categories almost exclusively contain images of women's clothing and no items in these categories are on our web application, our results showed that training on this additional data allowed our model to learn more meaningful features and improved its performance.

The availability of computational power also allowed us to train our model from scratch rather than use transfer learning, in which we would only train the last three layers of the model and all earlier layers were frozen with their weights set to the values obtained by training on the ImageNet dataset, accessed via the torchvision package. As noted in the above table, this provided a significant increase in performance but required the model to be trained for significantly longer before its performance stabilized (30 epochs versus 10 epochs).

All of these models were simultaneously trained on the classification and the in-shop retrieval benchmarks of the DeepFashion dataset respectively. The first model, which was trained on 20 categories of the classification benchmark was only trained on a subset of the in-shop retrieval benchmark corresponding to items of the "top" clothing type. All other models were trained on multiple clothing types in the classification benchmark and so, were trained and tested on the entirety of the in-shop retrieval benchmark.

We also aimed to tune hyperparameters of our model for optimal performance by performing runs with different hyperparameter values and logging metrics such as loss and top-k accuracy using Tensorboard. The following figures, which are screenshots from the Tensorboard dashboard, demonstrate how we used the tool to tune the learning rate by repeating training with different values of the learning rate and comparing performance across runs. Tensorboard logging also proved to be a valuable tool in monitoring the training of our models by observing their loss and *top-k* accuracy on the train and test sets. Once the test accuracy of the model was seen to remain constant/decrease across multiple epochs, we terminated training and

considered that run complete.

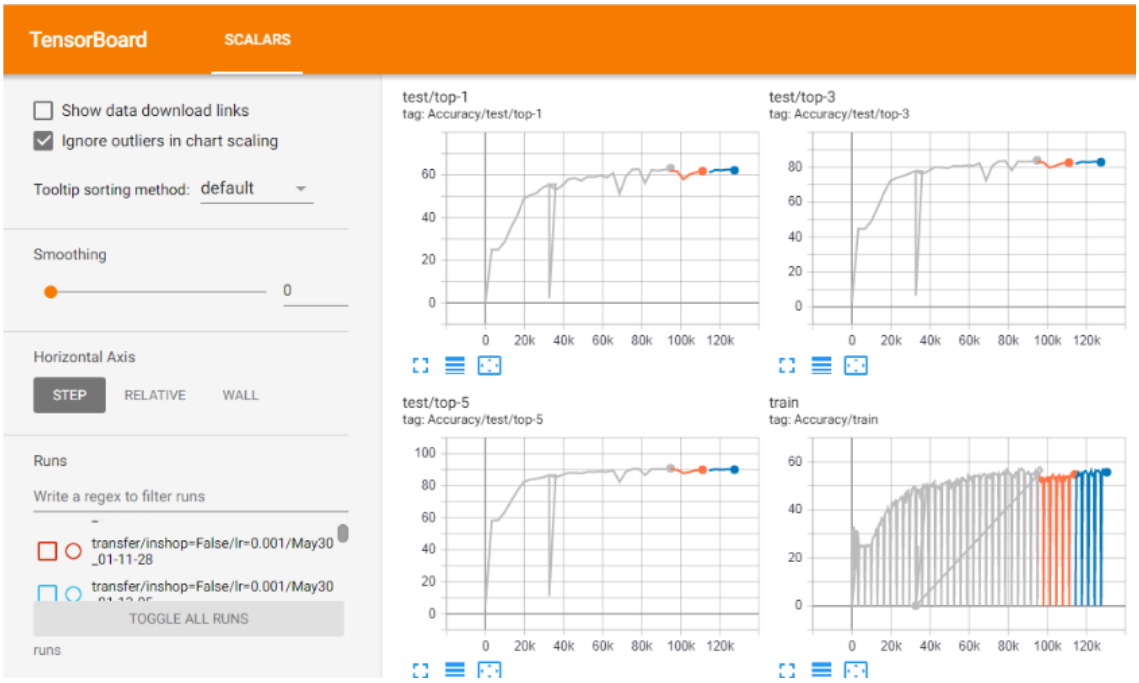

Figure 5.1: Logging model training of experiment #3 with TensorBoard

The above figure shows the log of model training of the best model of the third experiment. The model was trained for a total of 40 epochs, which took 26 hours. The x-axis represents steps taken by the model (each step is one batch). The model was tested after every epoch.

As expected, we see the test accuracy of the model increasing until it is almost constant. The model obtained after epoch 29 has the best test accuracy and so, is chosen as the best model. The train accuracy is logged after every 500 steps/batches, which explains the fluctuating nature of the 'Accuracy/train' graph. After applying smoothing to the graph, we see that the train accuracy is increasing with time, as expected. Note that the anomaly immediately before 40k steps is due to a power outage which interrupted training.

Although our initial aim was to optimize our model by tuning as many hyper-parameters as possible, this proved to be infeasible due to high training time of the model when it was trained and tested on the entirety of the DeepFashion classification and in-shop retrieval benchmarks due to the large size of the dataset. The classification benchmark contains 63,720 images while the in-shop retrieval bench-

mark contains a total of 54,642 images. Training the model for 40 epochs took approximately 26 hours. Hence, the optimization of hyperparameters has been left for future work. In this work we have trained with a small learning rate (0.001) to avoid overshooting optimal values and eliminate the need for hyper-parameter optimization.

As we will discuss in the following section, the increase in performance that resulted from these experiments was significant in our benchmarks but the difference in real world retrieval results in our store proved to be minimal. So any further improvement obtained by tuning hyper-parameters would result in a negligible increase in the real world performance of the tool. This is because the catalog of our store is scrapped from local men's e-commerce stores which have relatively small collections. We expect that with a larger and more diverse catalog, the increase in model performance would have translated to a more significant improvement in the real-world performance of the tool.

5.1 Results

The metrics used to test the performance of our models is the top-k accuracy in each respective task. For the classification task, we classify each image in the test set into one of the fifty categories. The classification is considered a success if the correct category label is among the top-k results of the model. Similarly, in-shop retrieval for an image in the test-query set is considered a success if on querying the test-gallery set, at least one of the top-k results is a photo of the same item.

The results of our experiment are summarised in the table below for precised understanding and in order to ease the process of comparison between the results of different experiment conducted as part of the training of the recommendation system. The table is provided as follows:

Results					
Model	**Number of Categories Trained On**	**Classification top-3 Accuracy (%)**	**Classification top-5 Accuracy (%)**	**In-shop. top-5 Retrieval Accuracy (%)**	**In-shop. top-20 Retrieval Accuracy (%)**
Experiment #1	20	87	95	74	86
Experiment #2	26	N.A.	N.A.	N.A.	N.A.
Experiment #3	50	84	91	81	91
FashionNet (from DeepFashion research paper)	50	82	90	68	76

Table 5.2: Accuracy Comparison

As earlier mentioned, we received access to Habib University's High Performance Computer while we were working on experiment 2. This significant increase in computational power meant that we could train our model on all 50 categories to obtain a strictly superior result and no longer needed to train our model on 26 categories. Hence, experiment 2 was abandoned.

Note that since the model in experiment 1 is only trained on 20 categories (all of which belong to the 'top' clothing type), for the classification benchmark it is only tested on a subset of the test set containing only items of these 20 categories. Similarly, experiment 2 is only tested on 26 categories. For the in-shop retrieval benchmark, both are tested on the entire test set.

Our best model significantly outperforms FashionNet, the architecture proposed by the creators of the DeepFashion dataset, on the in-shop retrieval benchmark. Since this dataset and the accompanying FashionNet model were published in 2016, they no longer represent the state-of-the-art. These benchmarks show that our model performs extremely well and is more than adequate for this business problem.

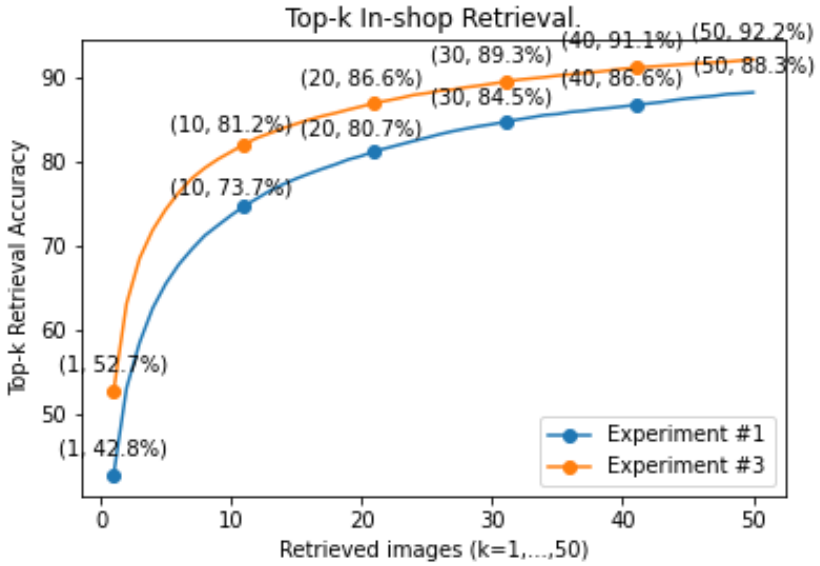

Figure 5.2: Top-k in-shop accuracy of experiments 1 and 3.

Although, the increase in performance that resulted from these experiments was significant in our benchmarks but the difference in real world retrieval results in our store proved to be minimal. This is because the catalog of our store is scrapped from local men's e-commerce stores which have relatively small collections. We expect that with a larger and more diverse catalog the increase in model performance would have translated to a more significant improvement in the real-world performance of the tool.

By matching the state-of-the-art in the field, we ensure that the performance of our model is not a limiting factor in taking this work forward and that it can be integrated into stores with larger and more complex catalogs while still performing more than adequately for this business scenario.

For the scope of this project, we do not believe that further optimization to our model (such as by tuning hyper-parameters or applying data augmentation) would yield any noticeable improvement in the tool's real world performance and so, the tuning of hyper-parameters is left for future work.

Figure 5.3: Results of AI Model - Tops category

Figure 5.4: Results of AI Model - Bottoms category

Figure 5.5: Results of AI Model - Eastern Wear

Figure 5.6: Results of AI Model - Eastern Wear

5.2 Complementary Recommendations

We have successfully extended the functionality of our AI model to also recommend complementary items. When the user uploads an image, they receive two sets of recommendations: items that are visually similar and items that are visually complementary to the outfit in the uploaded image. Items are said to be complementary if they can be paired to create an outfit.

When a user uploads a query image, the model is run on it and two pieces of information are extracted to perform complementary item recommendation:

- Predicted category of the item (t-shirt, jeans, hoodie etc.)
- Color vector- a set of 30 numbers that capture the color of the item

Based on the results of the category prediction by the model, a lookup table is then evaluated to find the categories of items that can be appropriately paired with it e.g. a t-shirt can not be paired with another t-shirt and hoodies and shorts are not generally worn together, so that combination is not recommended.

Currently, we have restricted the lookup table to the following 5 categories:

Category	**Complementary Categories**
T-shirt	Shorts, Jeans, Sweatpants, Hoodie, Jacket
Button-down	Hoodie, Jacket, Jeans, Joggers, Sweatpants
Jacket	Button-down, Sweater, T-shirt, Jeans, Sweatpants
Jeans	Button-down, Sweater, Tee, Tank, Jacket
Eastern	Waistcoats

In addition, the color vector is used to find the color of the outfit. This is done by comparing the vector to pre-computed color vectors of 10 solid colors and the closest matching color is found. The 10 colors that make up our database of solid colors can be seen in the following image.

A second lookup is then performed to find the colors that complement the color of the item. These color matching rules were created based on the outfits in our store which have been created.

The outputs of the clothing category compatibility and the color matching rules is used to fetch items that can be paired with the query item to create an outfit that is both appropriate and aesthetically appealing. The following are the snapshots of some of the outputs of the clothing category which are visually complimentary to the query image on the left.

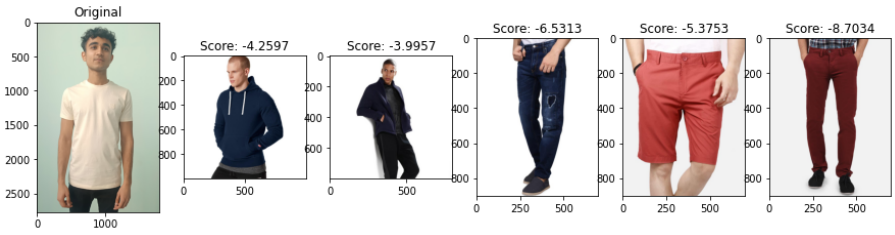

Figure 5.7: Results of AI Model for Complimentary Recommendation - Tops category

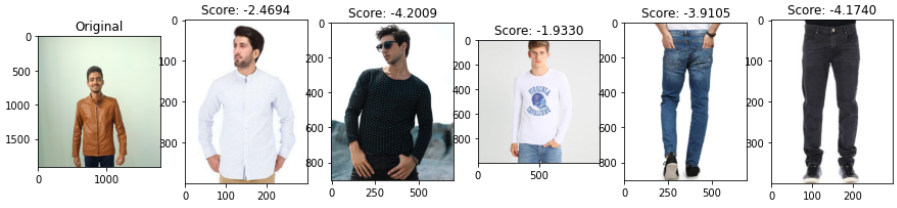

Figure 5.8: Results of AI Model for Complimentary Recommendations - Tops category

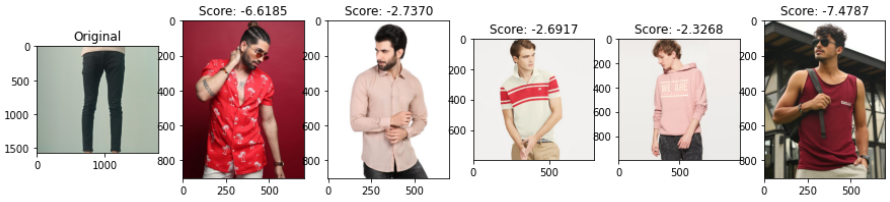

Figure 5.9: Results of AI Model for Complimentary Recommendations - Bottoms category

Figure 5.10: Results of AI Model for Complimentary Recommendations - Tops category (checked shirt)

Figure 5.11: Results of AI Model for Complimentary Recommendations - Two-Piece category

Figure 5.12: Results of AI Model for Complimentary Recommendations - Two-Piece category

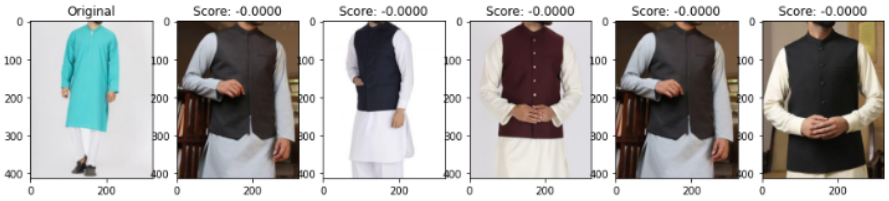

Figure 5.13: Results of AI Model for Complimentary Recommendations - Two-Piece category

6. Conclusion and Future Work

6.1 Conclusion

This thesis addressed the problem of product curation in fashion retail, i.e. selecting, organizing, and presenting the outfits in their best possible combinations, to the consumers. To solve this problem we presented a web-application that uses an AI model to recommend outfits to the shopper based on their input image.

The web application was developed using TypeScript and React at the client-side, while at the server-side, Python and Django were used. Whereas, React Apollo and Graphene Django were used for interacting with and creating the GraphQL API, respectively.

The web application uses a deep learning model to retrieve clothing items that are visually similar to those in the user uploaded image. The model uses a Convolutional Neural Network (CNN) to extract a feature vector of each item's image that is available in our store, which currently, consists of some of the local stores like J., Furor, Zellbury, Export Leftovers, and a dummy store. Similarly, a feature vector is extracted from the input image by the user and the items with the most similar feature vectors are displayed on our web-application. Our feature network consists of a ResNet50 model trained on the DeepFashion dataset - the largest and best-annotated dataset in the domain of fashion. Each image is then converted into a 512-dim feature vector, which captures the unique visual information of the clothing item in the image, including its style, color, and pattern. Clothing items whose feature vectors have minimum Euclidean distance from the query feature vector are then displayed to the user.

6.2 Future Work

In the future, first of all, we would look into the possibility to configure the recommendation model as a micro-service API and deploy the web application to a Web Server to make it feasible and accessible to both, the shoppers and the retailers.

Personalized Outfit Recommendation web application has a great potential to incorporate several enhancements in the future. Most importantly, refining the results of our AI model. Once the application is deployed and available publicly, there is a chance of collaborating with more local brands, meaning more data to feed into the AI model, consequently improving the recommendations. Additionally, the model will be trained in eastern fashion to make eastern recommendations more intelligent and robust, which is one of the novelties of this thesis. Furthermore, if the likes, dislikes, and purchase history of the customer are taken into account, it will further help to make the experience and recommendations more personalized for each customer. In addition to this, it can be further extended to the complete range of men's and women's clothing.

Lastly, a portable version of web application or a mobile application for Android and iOS would do a great deal in reaching out to a larger audience.

Appendix A. Data

Here is a dump of our 2TB data set. Enjoy!

Appendix B. Code

The entire code-base for all the modules - storefront, server, dashboard, this report, and the AI Model - can be found in this consolidated GitHub repository. Kaavish committee has already been given access to this private repository, and the codebase shall be made public for purposes of the evaluation and/or as per further instructions as given by the Kaavish committee.

The repository consists of all our work initiated and undertaken as part of the second part of our Kaavish module II. Detailed instructions on replicating the web application and a list of dependencies is also published there.

References

[1] Salesforce. *The Power of Personalized Shopping*. URL: https://www.salesforce.com/form/commerce/power-of-personalized-shopping.

[2] McKinsey. *How Retailers Can Keep Up with Consumers*. URL: https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers.

[3] The News Pakistan. *Pakistani Fashion Industry has Great Potential*. URL: https://www.thenews.com.pk/print/450217-pakistani-fashion-industry-has-great-potential.

[4] Digital Commerce 360. *US millennials now do most of their shopping online*. URL: https://www.digitalcommerce360.com/2019/03/26/millennials-online-shopping/.

[5] eMarketer. *Millennials Make More Apparel Purchases than Other Generations*. URL: https://www.emarketer.com/content/millennials-make-more-apparel-purchases-than-other-generations.

[6] Anand Tamrakar; Meenakshi Daga;Anusha Lunia. *Illustration on the superiority of Django over the other Python web frameworks*. URL: http://gujaratresearchsociety.in/index.php/JGRS/article/view/3109/2506.

[7] Tauqeer Hussain Tyler Crawford. *A Comparison of Server Side Scripting Technologies*. URL: https://csce.ucmss.com/cr/books/2017/LFS/CSREA2017/SER3291.pdf.

[8] Anand Malik. *What are the benefits of using react.js with Django?* URL: https://www.quora.com/What-are-the-benefits-of-using-react-js-with-Django#:~:text=js%20with%20Django%20%3F,-Thanks%20for%20A2A&text=Django%20is%20a%20back%2Dend,API%20to%20it%20as%20well..

[9] Jim A. Laredo Erik Wittern Alan Cha. *Generating GraphQL-Wrappers for REST(-like) APIs*. URL: https://arxiv.org/pdf/1809.08319.pdf.

[10] Binod Lama. *Implementing Graphql in Existing REST API*. URL: https://upcommons.upc.edu/bitstream/handle/2117/172870/memoria.pdf.

[11] Ziwei Liu et al. *DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations*. URL: https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Liu_DeepFashion_Powering_Robust_CVPR_2016_paper.pdf.

[12] A. Veit et al. *Learning visual clothing style with heterogeneous dyadic co-occurrences*. URL: https://www.cv-foundation.org/openaccess/content_iccv_2015/papers/Veit_Learning_Visual_Clothing_ICCV_2015_paper.pdf.

[13] J. McAuley et al. *Image-based recommendations on styles and substitutes*. URL: https://arxiv.org/pdf/1506.04757.pdf.

[14] X. Yi Y. Hu and L. S. Davis. *Collaborative fashion recommendation: A functional tensor factorization approach*. URL: http://delivery.acm.org/10.1145/2810000/2806239/p129-hu.pdf.

[15] Xufeng Han et al. *MatchNet: Unifying Feature and Metric Learning for Patch-Based Matching*. URL: https://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Han_MatchNet_Unifying_Feature_2015_CVPR_paper.pdf.

[16] Ziwei Liu et al. "DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations". In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.